DAS COLOUR-GENIE

ROM-LISTING

Klaus Kämpf



Das Colour-Genie ROM-Listing

Vorwort

Die Tatsache, daß Sie dieses Buch in der Hand halten, zeigt, daß Sie ein Colour-Genie Besitzer sind, der tiefer in die Geheimnisse der Maschinenprogrammierung einsteigen will.
Dabei ist ein kommentiertes Listing des Betriebssystems natürlich unerlässlich: zum einen zur Benützung der Routinen in eigenen Programmen, zum anderen um die internen Abläufe des Basics besser zu verstehen.

Um das Verständnis zu erleichtern, finden Sie vor dem eigentlichen dokumentierten ROM-Listing etliche Seiten wertvoller Erklärungen. Wenn Sie diese lesen, betrachten Sie ruhig schonmal die angegebenen Speicherteile im Listing.

Noch ein Hinweis:
Im Text entspricht das Paragraphenzeichen (§) dem "Klammeraffen" (@).
Ich hoffe, daß dieses Buch Ihnen eine wertvolle Hilfe sein wird und wünschen Ihnen weiterhin viel Spaß mit Ihrem Colour-Genie.

Krefeld, im Januar 1984

Klaus Kämpf



Speicheraufteilung

Das Basic teilt den ihm zur Verfügung stehenden Speicherplatz in insgesamt sieben Teile in folgender Reihenfolge auf:

1. Basic Programmtext
2. Basic Variablen
3. Freier Speicher
4. Zwischenspeicher (Stack)
5. Stringspeicher
6. SHAPE-Tabelle
7. Unterprogramme in Maschinensprache

Bis auf die SHAPE-Tabelle, sind die Größen der einzelnen Teile veränderbar bzw. abhängig vom jeweiligen Basicprogramm. Die Größe des Stringspeichers wird durch den CLEAR-Befehl festgelegt, die SHAPE- Tabelle ist auf 256 Bytes fixiert.

Die ersten beiden Teile 'wachsen' bei der Eingabe eines Programms (1. Teil) bzw. der Ausführung (2. Teil) nach oben. Die Teile 4 und 5 'wachsen' nach unten. Bei jeder Speicherzuweisung (Eingabe einer Programmzeile, neue Variablenzuweisung, neues CLEAR n etc.) wird die Obergrenze des Variablenteils mit der Untergrenze des Stacks verglichen (MEM ist die Differenz dieser beiden Grenzen). Sind weniger als 60 Bytes vorhanden, dann wird ein OM-Error erzeugt.

Der Stringspeicher wird ebenfalls von oben nach unten gefüllt. Die Differenz der Adresse des 'tiefsten' bzw. zuletzt eingefügten Strings (steht in 40D6H/40D7H) und der Startadresse des Stringspeichers wird bei jeder neuen Stringzuweisung errechnet. Ist die Länge des neuen Strings größer als diese Differenz, so wird der Stringspeicher neu umsortiert (vgl. 28DAH) und erneut getestet. Konnte kein Platz geschaffen werden, bricht die Routine mit einem OS-Error ab.

Das folgende Schema gibt einen Überblick über die Speicheraufteilung. In den in Klammern angegebenen Adressen befindet sich die jeweilige Grenzadresse. Die Grenzadresse zwischen dem Basicprogramm und den Variablen erhält man also durch

```
PRINT PEEK(&H40F9) + 256*PEEK(&H40FA)
```

und alle anderen Werte entsprechend.

Start des Speichers: 4800H oder 5800H

```
!=========================================! (40A4H)
!                                         !
!                                         !
!    Basic Programmtext                   !
!                                         !
!                                         !
!-----------------------------------------! (40F9H)
!                                         !
!                                         !
!    Einfache Variablen                   !
!                                         !
!                                         !
!- - - - - - - - - - - - - - - - - - - - -! (40FBH)
!                                         !
!                                         !
!    Indizierte (Feld-) Variablen         !
!                                         !
!                                         !
!-----------------------------------------! (40FDH)
!                                         !
!                                         !
!    Freier Speicher                      !
!                                         !
!                                         !
!-----------------------------------------! (40E8H)
!                                         !
!                                         !
!    Stack                                !
!                                         !
!                                         !
!-----------------------------------------! (40A0H)
!                                         !
!                                         !
!    Stringspeicher                       !
!                                         !
!                                         !
!-----------------------------------------! (40B1H)
!                                         !
!                                         !
!    SHAPE Tabelle                        !
!                                         !
!                                         !
!-----------------------------------------! TOPMEM
!                                         !
!                                         !
!    Unterprogramme in Maschinensprache   !
!                                         !
!                                         !
!=========================================!
```

Ende des Speichers: 7FFFH oder BFFFH

Ein- und Ausgabe

Alle Einheiten, die die Verbindung zum Benutzer herstellen (Tastatur, Bildschirm, Drucker etc.), werden Device genannt. Jedes Device wird von einer speziellen Routine (Treiberroutine) gesteuert, die jeweils ein einzelnes Zeichen verarbeitet. Zu jedem Device gehört ein Device Control Block (DCB), der nach einem festen Schema aufgebaut ist und es ermöglicht, den Aufruf einer Treiberroutine zu standardisieren.

Im Colour Genie stehen im RAM ab 4015H, für Tastatur, Bildschirm und Drucker jeweils ein DCB zur Verfügung. Jeder DCB besteht aus acht Bytes, die folgendermaßen belegt sind:

1. Byte:	DCB-Typ
2. und 3. Byte:	Adresse der Treiberroutine
4. bis 8. Byte:	Zur freien Verfügung der jeweiligen Treiberroutine

Der DCB-Typ gibt an, um welchen Typ von Device es sich handelt. Im Interpreter werden dabei drei Typen unterschieden:

01H: Nur Eingabe (z.B. Tastatur)
02H: Nur Ausgabe (z.B. Drucker)
04H: Ein- und Ausgabe (z.B. serielle Schnittstelle)

Will man ein Device ansprechen, muß im DE-Register nur die Adresse des gewünschten DCBs stehen. Nach Aufruf der entsprechenden Interpreterroutine (0013H, 001BH, 0023H) übernimmt das ROM die restliche Steuerung (Rettung der Register etc.).

Stimmt der DCB-Typ des angegebenen DCBs nicht mit der gewünschten Funktion überein, wird nicht die Treiberroutine angesprochen sondern nach 4033H im RAM gesprungen. Hier stehen 3 Bytes zur Aufnahme eines Sprungbefehls zur Verfügung und man kann einen eventuellen Fehler abfangen.

Soll nun z.B. der im Colour Genie fest installierte Druckertreiber geändert werden, muß nur die Adresse der neuen Treiberroutine im DCB stehen.
So kann man durch

```
POKE &H4026, PEEK(&H401EH) : POKE &H4027, PEEK(&H401FH)
```

alle Zeichen vom Drucker zum Bildschirm umleiten.

Wird die Treiberadresse der Tastatur und des Bildschirms auf die Adresse einer entsprechende Treiberroutine für die RS232 Schnittstelle geändert, so kann man das Colour Genie komplett von einem anderen Computer aus steuern.

Tastaturtreiber

Die Tastatur ist beim Colour Genie, im Gegensatz zu anderen Rechnern mit in den Speicherbereich einbezogen.
Bei vielen Computern übernimmt die Tastaturabfrage ein spezielles IC, daß dem Programm direkt den ASCII-Code der gedrückten Taste mitteilt. Bei solchen Hardwarelösungen ist der Code jeder Taste von vornherein auf den ASCII-Zeichensatz festgelegt und nicht zu ändern.
Der große Vorteil beim Colour Genie ist die softwaremäßige Tastaturkontrolle, bei der der Code jeder Taste vom Treiberprogramm festgelegt wird. Man erreicht dadurch nicht nur eine einfache Anpassung an andere Tastaturbelegungen (Graphikzeichen), sondern kann auch mehrere Tasten auf einmal abfragen und entsprechende Funktionen auslösen.

Damit auch beim schnellen Tippen keine Taste verlorengeht, arbeitet der Treiber nach dem n-key-rollover Prinzip. Dadurch wird erreicht, daß man die nächste Taste schon drücken kann, bevor man die letzte losgelassen hat.
Zu diesem Zweck werden bei jeder Tastaturabfrage die aktuellen Zeilenwerte in einer Tabelle im RAM abgelegt (4036H) und mit den letzten Werten verglichen. Sind beide Werte gleich, so wurde keine neue Taste gedrückt.
Für die Kontrolltasten, die nicht im ASCII-Standard erklärt sind, steht ab 0050H eine Tabelle mit den entsprechenden Werten.

Eine Besonderheit der Routine ist die spezielle Behandlung der BREAK-Taste. Wird BREAK gedrückt, gibt die Routine nicht direkt den Wert zurück, sondern führt vorher noch ein RST 28H aus. Bei 0028H steht ein Sprung ins RAM nach 400CH und dort ein RET. Es geschieht im Normalfall also nichts. Zum Schutz von Programmen kann man aber dadurch die BREAK-Taste sperren. Ändert man durch

```
POKE &H400C, &H00AF : POKE &H400D, &H00C9
```

den RET-Befehl in 400CH in ein XOR A mit nachfolgendem RET ab, so wird die BREAK-Taste nicht mehr erkannt. Ein gestartetes Basicprogramm kann also nicht mehr unterbrochen werden.

Im Interpreter stehen nun folgende Routinen zur Tastaturabfrage zur Verfügung:

002BH	Einmaliger Aufruf des Tastaturtreibers. (wie INKEY$)
035BH	wie 002BH, aber ohne Benutzung des DE-Registers
0049H	Warten auf Tastendruck. 002BH wird sooft aufgerufen, bis eine Taste gedrückt wurde.
0040H	Eingabe einer Zeile und Darstellung der gedrückten Zeichen auf dem Bildschirm (siehe 05D9H)

Bildschirmtreiber

Die Bildschirmanzeige stellt einen festgelegten Speicherbereich auf dem Bildschirm dar. Dabei wird zwischen dem Textspeicher (4400H bis 47FFH) und dem Grafikspeicher (4800H bis 57FFH) unterschieden. Je nach Programmierung des CRTCs (siehe Anhang A im Handbuch) und des Ports 255 (Anhang E) wird entweder Text oder Grafik auf dem Bildschirm ausgegeben.
Der Bildschirmtreiber ist nur für die Textdarstellung zuständig, die Grafikdarstellung übernehmen die Befehle PLOT, CIRCLE etc.

Bei der Textausgabe setzt sich das dargestellte Bild aus zwei Speicherbereichen zusammen. Das darzustellende Zeichen wird durch den Textspeicher (4400H bis 47FFH), seine Farbe durch den Farbspeicher (F000H bis F3FFH) bestimmt. Im Textspeicher stehen die ASCII-Codes der einzelnen Zeichen, im Farbspeicher die dazugehörigen Farbcodes.

Aufgabe des Bildschirmtreibers ist es, die auszugebenden Zeichen in den Textspeicher zu schreiben und den Farbspeicher entsprechend anzupassen. Die Erzeugung des Videosignals wird dann vom CRTC übernommen.
Eine Besonderheit beim Bildschirmtreiber ist noch die Möglichkeit ihn mit dem DCB-Typ 'Eingabe' aufzurufen. Ist in diesem Fall der Cursor eingeschaltet, so gibt der Treiber den ASCII-Wert des 'unter' dem Cursor befindlichen Zeichens zurück.

Im Bildschirm DCB befindet sich neben der Treiberadresse (30E4H) auch die aktuelle Cursoradresse. Auf dieser Adresse im Textspeicher steht der Cursor und dort erscheint das nächste auszugebende Zeichen. Des weiteren wird im DCB noch der ASCII-Code des Zeichens 'unter' dem Cursor und der aktuelle COLOUR-Wert abgespeichert. Dieser Wert entspricht jedoch nicht dem Wert im Farbspeicher (siehe 3563H im ROM-Listing).

Im Interpreter steht nur eine Routine zur Zeichenausgabe zur Verfügung:

0033H Ausgabe eines Zeichens auf dem Bildschirm
(wie PRINT CHR$(x))

Druckertreiber

Der Druckertreiber im ROM ist relativ kurz gehalten, da die meisten Drucker heutzutage über eigene Intelligenz verfügen. Man muß dem Drucker nur den ASCII-Wert des zu druckenden Zeichens übermitteln, die restliche Steuerung übernimmt das Gerät dann selbst.

Die einzige Steueraufgabe des Druckertreibers ist die Kontrolle der Seitenlänge. Man kann durch ein POKE &H4028,x, wobei x die Anzahl der Zeilen pro Druckerseite ist, den Seitenvorschub (ASCII-Code 0CH) durch den Druckertreiber kontrollieren lassen.

Der Druckeranschluß erfolgt über die beiden Ports des PSG (siehe Anhang B im Handbuch). Port A ist nur zur Ausgabe vorgesehen und übergibt das Zeichen zum Drucker. Port B hat zwei Aufgaben: Erstens muß er vom Drucker sogenannte Steuerbits übernehmen, mit denen der Drucker mitteilt ob er schon das nächste Zeichen übernehmen kann, oder noch beschäftigt ist. Die zweite Aufgabe des Ports B ist es, dem Drucker mitzuteilen wann ein Zeichen an Port A anliegt (Strobe).

Für die Druckerausgabe ist vom Interpreter nur eine Routine vorgesehen:

003BH Ausgabe eines Zeichens an den Drucker

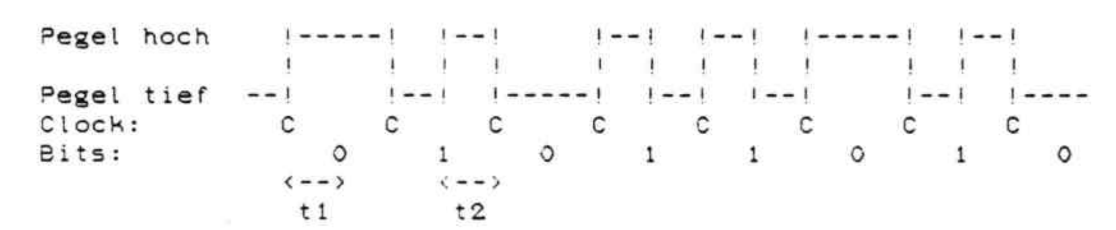
Der Druckertreiber kann nur Drucker mit paralleler Schnittstelle bedienen. Falls ein serieller Drucker angeschlossen werden soll, muß man eine entsprechende Treiberroutine schreiben und deren Adresse in den Drucker DCB schreiben.

Cassettenroutinen

Die Cassettenein- und ausgabe wurde nicht über das DCB-Schema realisiert, da die Cassettenoperationen nicht auf einzelenen Zeichen, sondern ganzen Datenblöcken basieren.
Ein solcher Datenblock ist entweder ein Programm oder durch einen PRINT#-1 Befehl entstanden.
Allen Datenblöcken vorangestellt wird ein sogenannter Leader und ein Sync-Byte mit deren Hilfe der Computer den Anfang eines Datenblocks erkennen kann. Der Leader besteht aus 255 mal dem Wert AAH und das Sync-Byte ist der Wert 66H (vgl. 023FH).

Jedes Byte, daß auf Cassette geschrieben werden soll, wird in seine acht Bits zerlegt und dann bitweise abgespeichert.
Zwischen je zwei Bits wird noch ein Synchronisationsimpuls (Clockimpuls) geschrieben, um eventuelle Gleichlaufschwankungen des Rekorders auszugleichen.
Auf der Cassette werden nur zwei verschiedenen Pegel abgespeichert und jedes Bit bzw. jeder Clockimpuls durch einen Pegelwechsel dargestellt. Fehlt zwischen zwei Clockimpulsen der Pegelwechsel, so ist das Bit '0', andernfalls '1'.

Das Byte 5AH (01011010) wird dann folgendermaßen abgespeichert:



Der Abstand zwischen einem Clockimpuls und dem Datenbyte (t1) wird von dem Wert in 4310H, der Abstand von diesem Datenbit zum nächsten Clockimpuls (t2) von dem Wert in 4311H bestimmt.

Das Lesen eines Bandes ist etwas schwieriger. Über den Leader und das Sync-Byte wird der Lesevorgang synchronisiert. damit nicht Clock- und Datenimpulse verwechselt werden. Beim Lesen eines Bytes wartet der Computer auf den (Clockimpuls) und führt dann eine Zeitschleife aus. Danach wird der aktuelle Pegel mit dem letzten Wert verglichen. Falls sich der Pegel geändert hat. ist das Bit gleich '1' ansonsten '0'. Die Wartezeit wird durch den Wert in 4312H bestimmt.

Ändert man die drei genannten Zeitwerte im gleichen Verhältnis. so können Cassettenoperation auch langsamer (bei schlechten Recordern) oder schneller (bei guten Recordern) gemacht werden.

Für Cassettenoperationen stehen folgende Routinen zur Verfügung:

023FH Leader und Sync auf Cassette schreiben
024CH Leader und Sync auf Cassette suchen
021FH Ein Byte auf Cassette schreiben
01EDH Ein Byte von Cassette lesen

Das Cassettenformat

Das Colour Genie kennt zwei verschiedene Cassettenformate: Das Format der SYSTEM-Bänder und das der CLOAD-Bänder. Ein Band, das vom CLOAD-Befehl gelesen werden soll, muß folgendes Format haben:

1. Leader und Sync
2. Ein ASCII-Zeichen für den Filenamen
3. Das Basicprogramm, wie es im Speicher steht
4. Drei Bytes 00H zur Endkennung

Aufzeichnungen von Maschinenspracheprogrammen (SYSTEM-Bänder) teilen sich in einzelne Datenblöcke auf. Zu jedem Datenblock wird seine Länge und die Blockadresse mit aufgezeichnet.

Ein Band, daß mit dem SYSTEM-Befehl geladen werden soll, muß folgendes Format haben:

1. Leader und Sync
2. Ein Byte 55H (SYSTEM-Kennung)
3. 6 Bytes Programmnamen
4. Blockkennung 3CH oder 78H. Ein Byte 3CH ist Kennung für einen Datenblock, 78H kennzeichnet das Aufzeichnungsende.
5. Datenblocks. Das erste Byte eines Datenblocks gibt die Länge des Blocks an. Danach folgt die Adresse, ab der die Daten des Blocks im Speicher abgelegt werden sollen. Nach den Datenbytes schließt eine Prüfsumme den Block ab. Diese Prüfsumme besteht aus einem Byte und ist die Summe aller Datenbytes und der beiden Adressbytes.
   Während des Ladevorgangs wird diese Summe vom SYSTEM-Befehl errechnet und am Blockende mit der gelesenen Prüfsumme verglichen. Sind beide Summen verschieden, so wird ein 'C' angezeigt, der Ladevorgang aber nicht unterbrochen.
6. Nach dem Byte 78H folgen die beiden Bytes der Einsprungadresse und der Ladevorgang wird beendet.

Graphikroutinen

Wie die den Cassettenoperationen, ist auch die Graphikausgabe auf einzelnen Routinen aufgebaut, die alle im ROM oberhalb 3000H liegen. Beim Aufruf dieser Routinen ist zu beachten, daß sie den Speicherbereich zwischen 42F0H und 4321H benutzen. Es stehen die folgenden Unterprogramme zur Verfügung:

38A9H	Umschaltung auf den FGR-Modus
38B0H	Umschaltung auf den LGR-Modus
3852H	FCLS (A = Farbcode)
3B8AH	PLOT (H = Y-Koordinate, L = X-Koordinate)
3CC6H	PLOT (wie 3B8AH, aber mit vertauschten Koordinaten)
3F3AH	A = CPOINT (L , H)
3C1FH	PLOT D,E TO H,L (Linie ziehen)

Die restlichen Befehle wie CIRCLE und PAINT holen sich ihre Parameter direkt aus dem Programmtext und können daher nicht so leicht von Maschinenspracheprogrammen benutzt werden.

Programmablauf nach dem Einschalten

Nach dem Einschalten des Colour Genies beginnt der Mikroprozessor mit der Programmausführung bei 0000H. Dort werden zuerst die Interrupts gesperrt, da der Speicher (DCBs etc.) noch nicht initialisiert ist.
Diese Grundeinstellung des Speicherbereicht 4000H bis 43FFH wird durch die Start-Routinen erreicht. Start 1 beginnt bei 0674H und schaltet den Bildschirm zuerst auf den Textmodus um. Danach werden die RST-Vektoren und DCBs aufgebaut. Dieser Aufbau wird viermal wiederholt, damit sich die Spannungsversorgung vollständig stabilisieren kann und der Speicher die Daten auch behält.
Die nächsten wichtigen Befehle folgen in Start 2 ab 056DH. Hier wird der Farbspeicher gelöscht und die Programmierungstabellen für den CRTC ins RAM geschrieben. Start 4 und 5 kopieren einen weiteren Teil des ROMs ins RAM und prüfen ob sich eine ROM-Cassette mit einem Basicprogramm im Slot befindet. Ist dies nicht der Fall, dann legt Start 6 (007BH) die Startadresse von Basicprogrammen fest. Diese ist normalerweise 5801H, also nach dem Grafikspeicher. Hält man jedoch während des Startvorgangs die MOD SEL-Taste gedrückt, so wird der gesamte Grafikspeicher für Programme freigehalten.
Falls eine ROM-Cassette mit einem Basicprogramm vorhanden ist, beginnt Start 6 bei 008BH und die Startadresse des Basicprogramms wird auf C001H festgelegt.
Danach wird der Speicherbereich von 41E5H bis 42E7H als Zeilenbuffer reserviert. Hier wird jede eingegebene Zeile zwischengespeichert und erst nach Drücken von RETURN übernommen.
Die folgenden Befehle sperren die sogenannten Disk Basic Vektoren (4152H bis 41E4H). Dies sind 'Ausgänge' aus dem normalen Basic und dienen zur Befehlserweiterung bei Anschluß eines Diskettenlaufwerks. Die Befehlsvektoren werden im Basic alle auf 'SN Error' gestellt und die restlichen Ausgänge durch 'RET' unwirksam gemacht. Will man einen Ausgang selbst nutzen, so braucht man nur den Vektor durch einen Sprungbefehl auf die eigene Routine zu ersetzen.
So kann z.B. durch folgende POKEs der Befehl LOAD anstelle von CLOAD benutzt werden: POKE &H4189, &H1F : POKE &H418A, &H2C.
Im folgenden wird dann der CRTC programmiert, das SP-Register gesetzt, nocheinmal auf ROM-Cassetten getestet und ein eventuell vorhandenes Maschinenspracheprogramm gestartet. Danach wird MEM SIZE abgefragt und dementsprechend die Speicherobergrenze festgelegt. Drückt man als Antwort nur 'RETURN', so wird der gesamte RAM-Speicher ab 4000H ausgetestet und die oberen 256 Bytes für die SHAPE-Tabelle freigehalten. Gibt man jedoch eine Speicheradresse an, so wird überprüft ob sie tatsächlich zum RAM-Speicher gehört und diese Adresse als Obergrenze des Speichers angenommen. Anhand dieses Wertes werden dann alle anderen speicherabhängigen Parameter errechnet, der Text 'COLOUR BASIC' ausgegeben und zum aktiven Befehlsmodus gesprungen.

Interne Abspeicherung

Jede eingegebene Programmzeile wird vom Interpreter in einen Zwischencode umgewandelt. Dadurch wird nicht nur der Speicherbedarf verringert, sondern auch die Ausführungszeit gekürzt.

Jeder Basicbefehl bekommt dabei ein bzw. zwei Bytes, sogenannte 'Token', zugeordnet.
Die meisten Basicbefehle werden intern durch einen Wert zwischen 128 und 251 repräsentiert. Die restlichen Befehle brauchen zwei Werte, wobei der erste 255 ist und der zweite zwischen 128 und 153 liegt.
Einen Überblick gibt eine Tabelle am Ende des Buches.

Jede Zeilennummer einer Programmzeile wird intern als 16 Bit Wert (2 Bytes) abgespeichert. Zeilennummer, die zu Befehlen gehören (z. B. GOTO 10), werden ziffernweise abgespeichert. Der Befehl GOSUB 2 braucht also 2 Bytes (1 Byte für den Befehl und 1 Byte für eine Ziffer), der Befehl GOSUB 10000 dagegen 6 (1 Byte für den Befehl und 5 Bytes für die 5 Ziffern).

Zu jeder Zeile wird zusätzlich die Endadresse der Zeile +1 abgespeichert. Dieser 'Link' (engl. Verbindung) zur nächsten Zeile ermöglicht es den Programmtext schneller nach einer bestimmten Zeile abzusuchen.
Das Ende jeder Zeile wird zusätzlich durch den Wert 00H gekennzeichnet.

Daraus ergibt sich für jede Zeile ein minimaler Speicherbedarf von 5 Bytes (2 Bytes Link, 2 Bytes Zeilennummer und 1 Byte Endkennung).

Als Beispiel soll nun die interne Darstellung des folgenden Programms dienen:
(vgl. Befehlstabelle und ASCII-Tabelle am Ende des Buches)

```
10 CLS
20 PRINT"Hallo !"
30 FGR:FCLS 1:LGR
40 GOTO 10
```

Dieses Programm steht so im Speicher:
(Alle Werte in Hexadezimal)

```
5801: 07 58  0A 00  84  00

5807: 16 58  14 00  B2 22 48 61 6C 6C 6F 20 21 22  00

5816: 25 58  1E 00  FF 85  3A  FF 87 20 31  3A  FF 86  00

5825: 2E 58  28 00  8D 20 31 30  00

582E: 00 00
```

Die Startadresse des Basicprogramms (hier 5801H) wird durch den Wert in den Speicherzellen 40A4H und 40A5H bestimmt. Die Endadresse plus zwei (hier 5830H) steht in den Speicherzellen 40F9H und 40FAH.

DDie erste Zeile (Zeile 10) besteht aus 6 Bytes. Die ersten beiden Bytes bilden den Link zur nächsten Zeile (= 5807H) die nächsten beiden ergeben die Zeilennummer (000AH = 10).
Zu beachten ist, daß immer das niederwertigte Byte vor dem höherwertigen Byte abgespeichert wird.
Nach der Zeilennummer folgt der Zwischencode der Zeile. Wie aus der Befehlstabelle hervorgeht, stellt das Token 84H den Befehl CLS dar. Das Zeilenende ist mit 00H gekennzeichnet.

In der nächsten Zeile wird nur das Befehlswort PRINT im Zwischencode dargestellt, der Text wurde Zeichen für Zeichen übernommen (vgl. ASCII-Tabelle).

Zeile 30 enthält Befehle, die sich auf die Grafik des Colour Genies beziehen. Diese Befehle brauchen zwei Token im Speicher (FF 85 = FGR, FF 87 = FCLS, FF 86 = LGR).
Die durch ein Leerzeichen vom Befehl FCLS getrennte 1 ist auch im Speicher durch den Wert 20H von den Token FF 87 getrennt. Es werden also bei der Zwischencodeerzeugung keine Leerzeichen gelöscht.

Der Link der letzten Zeile zeigt auf das Ende des Basicprogramms, daß durch einen Null-Link gekennzeichnet wird. Zusammen mit dem Zeilenende der letzten Zeile, steht dann 3 mal der Wert 00H im Speicher. Diese 3 aufeinanderfolgenden Nullen werden beim CLOAD-Befehl zur Erkennung des Programmendes verwendet.

Im Anschluß an den Programmtext werden die Variablen abgespeichert.
Jeder Variablenname kann aus ein oder zwei Zeichen, wobei das erste Zeichen ein Buchstabe sein muß, bestehen.
Ganzzahlige Werte brauchen 2, Werte einfacher Genauigkeit 4, Werte mit doppelter Genauigkeit 8 und Stringwerte 3 Bytes. Da Variablen ausser durch ihren Namen auch durch ihren Typ unterschieden werden, muß der Typcode mit abgespeichert werden.
Dieser Typcode wird auch gleichzeitig als Link zur nächsten Variablen benutzt. Addiert man nämlich zur Adresse des Typcodes den Wert des Typcodes + 3, so erhält man die Adresse der nächsten Variblen im Speicher.

Es ergibt sich also für ganzzahlige Variablen ein Speicherbedarf von 5 Bytes (1 Byte Typcode, 2 Bytes Namen, 2 Bytes Wert). Variablen einfacher Genauigkeit brauchen insgesamt 7, Variablen doppelter Genauigkeit 11 und Stringvariablen 6 Bytes.

Die Stringvariablen machen insofern eine Ausnahme, als daß ihr 'Wert' - also die Zeichenkette - nicht im Variablenspeicher, ondern im Stringspeicher steht. Die 6 Bytes Platzbedarf einer Stringvariablen spalten sich dabei in folgende Teile auf:
1 Byte Typcode (= 3), 2 Bytes Variablennamen, 1 Byte Länge der Zeichenkette und 2 Bytes Adresse der Zeichenkette im Stringspeicher bzw. (bei Stringkonstanten) im Programmtext.

Das folgenden Programm soll hier als Beispiel dienen:

```
10 B=100:B%=21
20 C3=27:D7#=1D+5
30 A$="KONSTANTE"


5801: 11 58  0A 00  42 D5 31 30 30  3A  42 25 D5 32 31  00

5811: 24 58  14 00  43 33 D5 32 37  3A  44 37 23 D5 31 44 CD
      35  00

5824: 34 58  1E 00  41 24 D5 22 53 54 52 49 4E 47 22  00

5834: 00 00

5836: 04  00 42  00 00 48 87

583D: 02  00 42  15 00

5842: 04  33 43  00 00 58 85

5849: 08  37 44  00 00 00 00 00 50 43 91

5854: 03  00 41  09  2C 58
```

Das Programm belegt den Speicher von 5801H bis 5833H.
In 5834H/5835H steht der Null-Link zur Kennzeichnung des Programmendes. Ab 5836H folgen die Variablen in der Reihenfolge wie sie im Programm auftauchen.
Als erstes steht der Typcode im Speicher. Daran anschließend die beiden Bytes des Variablennamens in umgekehrter Reihenfolge. Fehlt das zweite Zeichen des Namens, so wird 00H eingesetzt. Zum Schluß ist der Wert der Variablen in der internen Darstellung abgespeichert. Eine genaue Erklärung dieser Darstellungsart steht an anderer Stelle in diesem Buch.

Die Adresse einer Variable, die man durch den VARPTR-Befehl erhält, ist immer die Adresse des ersten Bytes des Wertes (also 5839H für B).
Die erste Variable hat keine expliziete Typangabe und wird daher mit einfacher Genauigkeit verarbeitet. Die zweite Variable hat zwar den selben Namen, unterscheidet sich aber durch die Typangabe von der ersten.

Bei der Stringvariablen am Ende ist nur die Länge (9 Zeichen) und die Adresse der zugehörigen Zeichenkette abgespeichert (582CH).

Bei Feldvariablen sieht die Speicherung etwas anders aus.
Ein Feld besteht immer aus Variablen gleichen Typs und gleichen Namens die durch ihren Index unterschieden werden. Daher werden Typcode und Namen eines Feldes nur einmal abgespeichert. Da bei Feldern der Typcode nicht als Link dienen kann, wird zusätzlich die Gesamtlänge des Feldes abgespeichert. Danach folgt die Anzahl der Dimensionen und dann für jede Dimension der maximale Index + 1 als 16Bit Wert. Am Schluß stehen die Werte der einzelnen Feldelemente mit jeweils 2 (Ganzzahlig), 4 (Einfache Genauigkeit), 8 (Doppelte Genauigkeit) oder 3 Bytes (Strings).
Die genaue Reihenfolge erhält man, wenn man die Indizes der Reihenfolge nach durchzählt. Für das Feld A(2,1) ergibt sich also folgende Reihenfolge:
A(0,0), A(1,0), A(2,0), A(0,1), A(1,1), A(2,1)

Eingabe und Ausführung von Befehlen und Programmen

Die Eingabe von Befehlen und Programmzeilen geschieht im sogenannten aktiven Befehlsmodus. Dieser Programmteil des ROMs (ab 1A19H) übernimmt die Zeileneingabe und die Abspeicherung neuer Programmzeilen.
Jede eingegebene Zeile wird, falls die Eingabe nicht mit BREAK beendet wurde, in den Zwischencode umgewandelt (1BC0H ff).
Steht am Anfang der Befehlszeile keine Zeilennummer, dann wird die Zeile direkt ausgeführt ansonsten ins Programm übernommen

Die Ausführung von Basicprogrammen und Befehlen läuft über die Programmschleife (1D1EH). Das HL-Register enthält immer die Adresse des nächsten auszuführenden Befehls bzw. Tokens. Im ROM-Listing ist diese Adresse mit PTZ ('ProgrammTextZeiger') bezeichnet. Aus dem Tokenwert und der Sprungadressentabelle wird dann die Adresse der Befehlsroutine ermittelt.

Jede Routine erhält die Adresse ihrer Parameter im HL-Register mitgeliefert und muß dafür sorgen, daß diese abgearbeitet werden. Bei der Rückkehr in die Programmschleife muß der PTZ auf das Befehls- bzw. Zeilenende zeigen.

Der aktuelle PTZ wird (vor der Befehlsausführung) im RAM bei 40E6H/40E7H abgelegt. Dadurch weiß die RESUME-Routine, ab wo der letzte Befehl wiederholt werden muß. Der aktuelle Stackpointer wird bei 40E8H/40E9H abgespeichert. Der Bereich oberhalb der Stackpointeradresse ist das sogenannte Basic-Stack. Hier werden von GOSUB und FOR die Parameter für RETURN und NEXT abgespeichert.

Diese Parameter werden folgendermaßen im Stack abgelegt:

GOSUB:
(vgl. GOSUB ab 1EB1H ff und RETURN ab 1EDEH)

(SP + 04H) : LSB des PTZ
(SP + 03H) : MSB des PTZ
(SP + 02H) : LSB der aktuellen Zeilennummer
(SP + 01H) : MSB der aktuellen Zeilennummer
(SP + 00H) : 91H (GOSUB-Kennung)

Der PTZ ist die Adresse, ab der die Programmausführung bei RETURN fortgesetzt werden soll.

FOR-Schleife mit einer Variablen einfacher Genauigkeit:
(vgl. 1CA1H ff)

```
(SP + 10H) : LSB des PTZ
(SP + 0FH) : MSB des PTZ
(SP + 0EH) : LSB der aktuellen Zeilennummer
(SP + 0DH) : MSB der aktuellen Zeilennummer
             Endwert der Schleife:
(SP + 0CH) : LSB
(SP + 0BH) : LSB
(SP + 0AH) : MSB
(SP + 09H) : EXP
             Schrittweite der Schleife:
(SP + 08H) : LSB
(SP + 07H) : LSB
(SP + 06H) : MSB
(SP + 05H) : EXP
(SP + 04H) : 01H (= VT - 3)
(SP + 03H) : SGN der Schrittweite (FFH, 00H oder 01H)
(SP + 02H) : LSB der Adresse der Schleifenvariablen
(SP + 01H) : MSB der Adresse der Schleifenvariablen
(SP + 00H) : 81H (FOR-Kennung)
```

FOR-Schleife mit einer Integervariablen

```
(SP + 10H) : LSB des PTZ
(SP + 0FH) : MSB des PTZ
(SP + 0EH) : LSB der aktuellen Zeilennummer
(SP + 0DH) : MSB der aktuellen Zeilennummer
(SP + 0CH) : LSB des Endwerts
(SP + 0BH) : MSB des Endwerts
(SP + 0AH) : LSB der Schrittweite
(SP + 09H) : MSB der Schrittweite
(SP + 08H) : Unbenutzt
(SP + 07H) : Unbenutzt
(SP + 06H) : Unbenutzt
(SP + 05H) : Unbenutzt
(SP + 04H) : FFH (= VT - 3)
(SP + 03H) : SGN der Schrittweite (FFH, 00H oder 01H)
(SP + 02H) : LSB der Adresse der Schleifenvariablen
(SP + 01H) : MSB der Adresse der Schleifenvariablen
(SP + 00H) : 81H (FOR-Kennung)
```

Der PTZ zeigt auf den ersten Schleifenbefehl, die aktuelle Zeilennummer ist die Nummer dieser Befehlszeile.

Das interne Variablenformat

Ganze Zahlen

Eine Speicherzelle im Colour Genie sind immer 8 Bits oder kurz 'Byte'. Der höchste Wert, den man mit einem Byte darstellen kann ist 11111111 (= 255). Da die Befehle PEEK und POKE immer auf eine Speicherzelle (ein Byte) zugreifen, können nur Werte zwischen 0 und 255 angegeben werden.
Nimmt man nun zwei hintereinanderliegende Speicherstellen zusammen, so können mit den zwei Bytes (16 Bits), Werte zwischen 0 und 65535 dargestellt werden.-
Die unteren 8 Bits einer solchen 16 Bit Zahl stehen immer zuerst im Speicher. Soll also z.B. die Binärzahl 0101101011000011 in den Speicherzellen 20000 und 20001 abgespeichert werden, erhält die erste Zelle 11000011 und die zweite 01011010.

Auf diese Weise werden im Colour Genie die Zeilennummern abgespeichert.
Wieso dann nicht Zeilennummern bis 65535 erlaubt sind, liegt daran, daß z.B. die Zeilennummern 65534 und 65535 anders verwendet werden. So bekommt jeder Befehl der ohne Zeilennummer - also zur direkten Ausführung - eingegeben wird, die Zeilennummer 65535 zugeordnet. So kann die Fehlerroutine entscheiden, ob bei einer Fehlermeldung die Zeilennummer mit angegeben wird oder nicht. Während der MEM SIZE Abfrage ist die Zeilennummer auf 65534 festgelegt. In diesem Fall springt die Fehlerroutine zur MEM SIZE Abfrage zurück.

Zur Darstellung von negativen Werten, wird im Colour Genie die Anzahl der Bits einer Integerzahl auf 15 beschränkt und das 16. Bit (höchstes Bit) als Vorzeichen genommen.
Die Binärzahl 0111111111111111 (Eine 0 und 15 Einsen) hat dann den Wert +32767. Setzt man das 16. Bit auf '1', so erhält man eine negative Nummer.
Aber 1111111111111111 (16 Einsen) im Integerformat ist nicht -32767, sondern -1. Es gibt zwei Wege den Wert einer solchen Integerzahl zu errechnen:
Falls das Vorzeichenbit '1' ist, so ermittelt man den Wert der 16 Bits, im obigen Beispiel ergibt dies 65535 (16 Einsen). Von diesem Wert zieht man dann 65536 ab und erhält den richtigen Wert (hier -1). Ist das Vorzeichenbit 0, dann geben die verbleibenden 15 Bits den richtigen Wert an.
Die zweite Methode ist etwas komplizierter. Wenn das 16. Bit gleich '0' ist, so bleibt alles beim alten. Ist es aber '1', so vertauscht man in restlichen 15 Bits, alle '1' mit '0' und umgekehrt. Dann wird eins addiert und man erhält den positiven Wert der (negativen) Integerzahl.

Einfache und doppelte Genauigkeit (Fließkommaformat)

Das Integerformat erlaubt nur die Verarbeitung ganzer Zahlen. Zur Darstellung von Brüchen im Binärsystem führt man nun einen 'Binärpunkt' ein.
Im Integerformat verdoppelt sich der Stellenwert der einzelnen Binärziffern von rechts nach links. Bei der Zahl 111 hat die erste Ziffer von rechts den Stellenwert 1, die nächste Ziffer links daneben erhält den doppelten Stellenwert (2) usw.
Geht man von links nach rechts vor, so halbiert sich der Stellenwert jeweils. Rechts der Einerstelle steht dann der Stellenwert 1/2, dann 1/4, 1/8 usw. Zur Kennzeichnung der Einerstelle schreibt man dann rechts davon einen 'Binärpunkt'.
Die Binärzahl 101.011 hat dann den Wert 5.375 (4+1+1/4+1/8).

Da der Binärpunkt aber nicht mit abgespeichert werden kann, ändert man die Schreibweise von 101.011 in 0.101011 * 2 hoch 3. Die Zahl wird also in die sogenannte Mantisse (0.101011) und den Exponenten (2 hoch 3) aufgespalten.
(1/2 + 1/8 + 1/32 + 1/64 * 8 = 5.375)

Zur Erhöhung der Genauigkeit werden alle Zahlen mit einer möglichst großen Anzahl von Nachkommastellen abgespeichert.
Der Binärpunkt wird dazu soweit nach rechts verschoben, bis alle Nullen zwischen dem Binärpunkt und der ersten '1' gelöscht sind.
Die Binärzahl 0.00101 * 2 hoch 3 wird als 0.10100 * 2 hoch 1 gespeichert.

Für die Zahl 11.4375 ergeben sich demnach folgende Darstellungsmöglichkeiten:

1. 001011.0111
   (8 + 2 + 1 + 1/4 + 1/8 + 1/16 = 11.4375)

2. 0.0010110111 * 2 hoch 6
   (1/8 + 1/32 + 1/64 + 1/256 + 1/512 + 1/1024 * 64 = 11.4375)

3. 0.1011011100 * 2 hoch 4
   (1/2 + 1/8 + 1/16 + 1/64 + 1/128 + 1/256 * 16 = 11.4375)

Da bei der 3. Möglichkeit das erste Bit nach dem Binärpunkt immer eine '1' ist, braucht dieses Bit nicht mit abgespeichert zu werden.
Diese Tatsache wird im Colour Genie zur Vorzeichenkennung genutzt. Vor der Speicherung wird diese '1' gelöscht und eine '0' für positives bzw. '1' für negatives Vorzeichen eingesetzt.
Die Zahl +11.4375 ist dann gleich 0.0011011100 * 2 hoch 4 und die Zahl -11.4375 gleich 0.1011011100 * 2 hoch 4.

Von den 4 Bytes einer Zahl einfacher Genauigkeit stehen 3 Bytes für die Matisse zur Verfügung, die 24 Bits lang ist. Das vierte Byte enthält den (Zweier-)Exponenten. Da der Exponent sowohl positiv als auch negativ sein kann, muß ein Bit des Exponentenbytes für das Vorzeichen reserviert werden. Dem eigentlichen Exponenten stehen dann 7 Bits zur Verfügung. Zusätzlich wurde noch festgelegt, daß eine Zahl den Wert 0 hat, falls das Exponentenbyte Null ist.
Dies hat den Vorteil, daß die Prüfung auf Null auf ein Byte beschränkt bleibt.
Im Exponentenbyte steht nun der Zweierexponent +128. Dem Exponenten 129 entspricht also eine Multiplikation der Mantisse mit 2 (= 2 hoch (129 - 128)), dem Exponenten 126 eine Multiplikation mit 0.25 (= 2 hoch (126 - 128)).

Die größte Zahl, die auf diese Weise dargestellt werden kann, ist 2 hoch 127 = 1.701411 * 10 hoch 38 = 1.701411E38.

Die doppelte Genauigkeit wird nach dem selben Schema aufgebaut, nur daß für die Mantisse 7 Bytes (= 56 Bits) zur Verfügung stehen.

Wie beim Integerformat auch, werden die 4 bzw. 8 Bytes einer Fließkommazahl in umgekehrter Reihenfolge im Speicher abgelegt. Das unterste Byte der Mantisse, das LSB (engl. Least significant Byte, niederwertigstes Byte) steht zuerst im Speicher. Vor dem Exponenten, der als letztes abgespeichert wird, steht das höchste Byte (engl. Most significant Byte, kurz MSB, höchstwertigstes Byte) der Mantisse mit dem Vorzeichenbit (vgl. Seite 97 im Handbuch).

Zur Verdeutlichung des Fließkommaformats sollen nun die folgenbeiden Beispiele dienen.

Angenommen, die Variable A hat den Wert 10. Über den VARPTR-Befehl erhält man die Adresse der Variablen und ein PEEK auf diese und die nächsten drei Adressen ergibt vier Werte in folgender Reihenfolge:

```
  0  0  32  132 oder binär 00000000 00000000 00100000 10000100
```

Die ersten drei Werte sind die Mantisse in umgekehrter Reihenfolge, das vierte der Exponent. Zieht man vom Exponenten 128 ab, so ergibt dies einen Exponenten von 4, die Mantisse muß also mit 2 hoch 4 (= 16) multipliziert werden. Die Mantisse in der richtigen Reihenfolge und einer '1' statt des Vorzeichenbits ist dann:

```
  .10100000 00000000 00000000 = 1/2 + 1/8 = 0.625
```

Mit 16 multipliziert ergibt das genau 10.

Bei einer anderen Variablen erhält man über VARTPTR und PEEK die Werte 0, 208, 201 und 131. Welcher Wert wurde der Variablen zugewiesen ?

Der Exponent ist 3 (= 131 - 128) und die Mantisse in der richtigen Reihenfolge ist 11001001 11010000 00000000. Die '1' im höchsten Bit zeigt, daß die Zahl negativ ist.
Die Errechnung der Mantisse sieht dann folgendermaßen aus:
(Statt des Vorzeichenbits wurde wieder eine '1' eingesetzt)

```
 .1 1 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
                           ------------------------> 1/4096
                       ----------------------------> 1/1024
                     ------------------------------> 1/512
                   --------------------------------> 1/256
             --------------------------------------> 1/32
       --------------------------------------------> 1/4
   ------------------------------------------------> 1/2
                                                     ------
                                                   = .78833
```

Mit Zweierexponenten und Vorzeichen ergibt dies einen Wert von -6.30664

Zeichenketten (Strings)

Im Gegensatz zu Zahlenwerten, teilt sich bei Strings die Abspeicherung in zwei Teile. Der erste Teil, der sogenannte Stringvektor, besteht aus 3 Bytes und enthält die Länge der Zeichenkette sowie ihre Adresse im Speicher. Der zweite Teil ist die Zeichenkette selbst, die meistens im Stringspeicher steht.
Über den VARPTR-Befehl erhält man nur die Adresse des ersten Teils, nicht die Adresse der Zeichenkette. Ein PEEK auf dieser und den beiden nächsten Adresse ergibt dann als ersten Wert die Stringlänge und als zweiten und dritten Wert die Adresse der eigentlichen Zeichenkette (auch hier stehen wieder die unteren 8 Bits der 16Bit Adresse zuerst im Speicher).
Angenommen man will wissen wo die Zeichenkette der Stringvariablen A$ im Speicher steht, so erhält man die Adresse auf folgende Weise:

```
PRINT PEEK( VARPTR(A$)+1 )  +  PEEK( VARPTR(A$)+2 ) * 256
```

Die Rechenroutinen des Colour Genies

Wie im Mikroprozessor auch, laufen alle Rechnungen im Colour Genie Basic über einen 'Akkumulator'. Über diesen Zwischenspeicher werden alle Argumente an die Rechenroutinen übergeben, die ihrerseits das Ergebnis auch wieder dort ablegen.

Im Colour Genie gibt es zwei solcher Zwischenspeicher genannt X und Y (411DH ff und 4127H ff).
Der Zwischenspeicher X wird für alle Funktionen mit einem Argument (SIN, COS , INT, SQR etc.) verwendet und enthält auch grundsätzlich das Ergebnis. Der Zwischenspeicher kann alle vier Variablenformate aufnehmen. Welches Format sich gerade im Zwischenspeicher X befindet, wird durch den Variablentyp (VT) an der Stelle 40AFH bestimmt. Der VT ist als die Anzahl der Bytes, die den Wert der Variablen enthalten, festgelegt.
Der VT eines Integer- wertes ist also 2 (Eine Integerzahl besteht aus 2 Bytes), der eines Wertes einfacher Genauigkeit 4 (Das Singleformat braucht 4 Bytes) und der VT für die doppelte Genauigkeit ist 8 (Das Doubleformat braucht 8 Bytes).

Die Stringfunktionen benutzen ebenfalls den Zwischenspeicher X und belegen ihn aber mit der Adresse des Stringvektors.

Soll nun z.B die Wurzel einer SNG-Zahl errechnet werden, so muß man die 4 Bytes in den Zwischenspeicher X übertragen, den VT auf 4 setzen und kann dann die SQR-Routine aufrufen. Das Ergebnis steht dann ebenfalls wieder in X.

Bei allen Routinen mit zwei Argumenten (Grundrechenarten und das Potenzieren) ist die Werteübergabe anders geregelt.
Im Integerformat werden die beiden 16Bit Werte in den Prozessorregistern DE und HL übergeben. Im Singleformat wird das eine Argument in X und das andere in den beiden Prozessorregistern BC und DE übergeben. B enthält dabei den Exponenten, C das MSB der Mantisse und DE die beiden anderen Mantissenbytes. Bei doppelter Genauigkeit wird der Zwischenspeicher Y für die Aufnahme des zweiten Arguments benutzt. In allen Fällen befindet sich nach der Ausführung der Rechenroutine das Ergebnis wieder im Zwischenspeicher X.

Im ROM stehen nun folgende Rechenroutinen zur Verfügung:

(X und Y bezeichnen den jeweiligen Zwischenspeicher, BCDE die beiden Register BC und DE, (HL) die Adresse des Wertes und (SP) daß sich der Wert im Stack befindet)

Umwandlungsroutinen zur Ein- und Ausgabe

Jeder Zahlenwert muß von der 'externen' in die 'interne' Darstellung verwandelt werden. Eine vom Benutzer eingegebene Zahl sieht der Computer nämlich nicht als Dezimalzahl, sondern als die Werte der einzelnen Ziffern nach dem ASCII-Code. Die '1' einer Zahl stellt demnach für den Computer nicht den Wert 1 dar, sondern den Wert des Zeichens '1', also 48 (vgl. ASCII-Tabelle).
Für die Umwandlung vom externen (ASCII-)Format zum internen Binärformat stehen im ROM verschiedene Routinen zur Verfügung.

Für die Eingabe:

0E6CH	Umwandlung einer Zeichenkette bestehend aus Ziffernzeichen, Vorzeichen ('+' oder '-'), Dezimalpunkt, Exponentenangabe ('E' oder 'D') und einer Typangabe ('%', '!' oder '#') Ergebnis in X
1E5AH	Umwandlung einer Zeichenkette, Ergebnis im HL-Register.(Für Zeilennummer und MEM SIZE-Abfrage)

Für die Ausgabe:

0FAFH	Ausgabe des Wertes im HL-Register (0 - 65535) (Für Zeilennummern)
0FBDH	Umwandlung von X in einen unformatierten String ab 4130H (Für normales PRINT und STR$)
0FBEH	Umwandlung von X in einen formatierten String ab 4130H (Für PRINT USING)

Durch einen CALL 28A7H im Anschluss an die Umwandlungsroutinen 0FBDH und 0FBEH wird der erzeugte String ausgegeben.

Weitere Hilfsroutinen

Adresse der Routine	Funktion
09A4H	(SP) = X (SNG) X in den Stack retten
09B1H	X = BCDE = (HL) (SNG) X und BCDE mit den ab (HL) stehenden 4 Bytes laden
09B4H	X = BCDE (SNG) BCDE in X abspeichern
09BFH	BCDE = X (SNG) BCDE mit X laden
09C2H	BCDE = (HL) (SNG) BCDE mit den ab (HL) stehenden 4 Bytes laden
09CBH	(HL) = X (SNG) X in (HL) bis (HL + 3) abspeichern
09F4H	X = Y (DBL)
09F7H	X = (HL) (SNG oder DBL je nach VT) X mit den ab (HL) stehenden 4 oder 8 Bytes laden
09FFH	(HL) = X (SNG oder DBL je nach VT) X in (HL) bis (HL + 3 (7)) abspeichern
0A9AH	X = HL (INT) Kopiert den Integerwert in HL nach X

Mathematische Funktionen

Adresse der Routine	Ausgeführte Rechnung	Genauigkeit des Arguments	Genauigkeit des Ergebnisses
0977H	X = ABS (X)	INT, SNG, DBL	
15BDH	X = ATN (X)	INT, SNG, DBL	SNG
0ABDH	X = CDBL (X)	INT, SNG, DBL	DBL
0A7FH	X = CINT (X)	INT, SNG, DBL	INT
0AB1H	X = CSNG (X)	INT, SNG, DBL	SNG
1541H	X = COS (X)	INT, SNG, DBL	SNG
1439H	X = EXP (X)	INT, SNG, DBL	SNG
0B26H	X = FIX (X)	INT, SNG, DBL	
0B37H	X = INT (X)	INT, SNG, DBL	
0809H	X = LOG (X)	INT, SNG, DBL	SNG
14C9H	X = RND (X)	INT, SNG, DBL	SNG
098AH	X = SGN (X)	INT, SNG, DBL	INT
1547H	X = SIN (X)	INT, SNG, DBL	SNG
13E7H	X = SQR (X)	INT, SNG, DBL	SNG
15A8H	X = TAN (X)	INT, SNG, DBL	SNG

Test- und Vergleichsroutinen
(CP bedeutet Compare wie im Z80 Assembler)

Adresse der Routine	Operation	Genauigkeit des Arguments	Ergebnis
0020H	Testet den VT		
0AD9H	VT auf INT setzen		
0AEFH	VT auf SNG setzen		
0AECH	VT auf DBL setzen		
0994H	A = SGN (X)	INT, SNG, DBL	A-Register
099BH	A = SGN (X)	INT	A-Register
0955H	A = SGN (X)	SNG, DBL	A-Register
0018H	CP HL , DE	-- (Register)	A-Register
0A0CH	CP X , BCDE	SNG	A-Register
0A39H	CP HL , DE	INT	A-Register
0A4FH	CP X , Y	DBL	A-Register
0A78H	CP Y , X	DBL	A-Register

Grundrechenarten

Adresse der Routine	Ausgeführte Rechnung	Genauigkeit
0B2DH	X = DE + HL	INT
0BC7H	X = DE - HL	INT
0BF2H	X = DE * HL	INT
2490H	X = DE / HL	INT
0716H	X = BCDE + X	SNG
0713H	X = BCDE - X	SNG
0847H	X = BCDE * X	SNG
08A2H	X = BCDE / X -	SNG
0C77H	X = X + Y	DBL
0C70H	X = X - Y	DBL
0DA1H	X = X * Y	DBL
0DE5H	X = X / Y	DBL
093EH	X = X * 10	SNG
0E4DH	X = X * 10	DBL
0F0AH	X = X * 10	SNG, DBL
0897H	X = X / 10	SNG
0DDCH	X = X / 10	DBL
0F18H	X = X / 10	SNG, DBL
0708H	X = X + 1/2	SNG
070BH	X = (HL) + X	SNG
0710H	X = (HL) - X	SNG
08A0H	X = (SP) / X	SNG
097BH	X = -X	INT, SNG, DBL

Verwendete Schreibweise

A,B,C,D,E H,L,BC,DE, HL,IX,IY, SP,PC	Bezeichnung der entsprechenden Z80-Register
BCDE	Zusammenfassung des BC und DE Registers zur Aufnahme eines Wertes einfacher Genauigkeit
X	Zwischenspeicher X (ab 411DH)
Y	Zwischenspeicher Y (ab 4127H)
Exp (X)	Exponent des Wertes im Zwischenspeicher X
Exp (Y)	Exponent des Wertes im Zwischenspeicher Y
Exp	Exponent eines Fließkommawertes
Sign	Vorzeichen
VT	Variablentyp (02 = INT, 03 = STR, 04 = SNG, 08 = DBL)
ZP	Zeilenpointer, (= Link)
ZN	Zeilennummer
PTZ	Programmtextzeiger
A= B	A ist gleich B
A: B	A erhält die Adresse von B (A 'zeigt' auf B)
A.n	Bit n des Registers A (andere Register entsprechend)
I:	Liste der Eingabeparameter
O:	Liste der Ausgabeparameter
UPRO	Unterprogramm
--	Der Befehl ist unbenutzt
'*'	Das folgende Byte ist doppelt verwendet

```
; Start des BASIC-Interpreter ROMs
; Einsprung bei 0000H nach Einschalten des Geräts oder nach <RESET>-<R>
; Kaltstart

  0000 F3             DI                 ;Interrupts sperren
  0001 AF             XOR     A          ;A=0
  0002 C37406         JP      0674H      ;Weiter bei 0674H

  0005 C30040         JP      4000H      ;--


  0008 C30040         JP      4000H      ;RST 08H: Sprung nach 4000H und
                                         ;von dort weiter bei 1C96H

  000B E1             POP     HL         ;--
  000C E9             JP      (HL)

  000D C30000         JP      0000H      ;--


  0010 C30340         JP      4003H      ;RST 10H: Sprung nach 4003H und
                                         ;von dort weiter bei 1D78H

; DCB-Aufruf Eingabe (AF,DE)
; I: DE = DCB-Adresse
; O: A  = Eingegebenes Byte

  0013 C5             PUSH    BC         ;BC retten
  0014 0601           LD      B,01H      ;DCB-Typ: Eingabe
  0016 182E           JR      0046H      ;und DCB aufrufen


  0018 C30640         JP      4006H      ;RST 18H: Sprung nach 4006H und
                                         ;von dort weiter bei 1C90H

; DCB-Aufruf Ausgabe (AF,DE)
; I: DE = DCB-Adresse
;     A  = Auszugebendes Byte
; O: -

  001B C5             PUSH    BC         ;BC retten
  001C 0602           LD      B,02H      ;DCB-Typ: Ausgabe
  001E 1826           JR      0046H      ;und DCB aufrufen


  0020 C30940         JP      4009H      ;RST 20H: Sprung nach 4009H und
                                         ;von dort weiter bei 25D9H

; DCB-Aufruf Ein- und Ausgabe (AF,DE) (unbenutzt)
; I: DE = DCB-Adresse
; O: ?

  0023 C5             PUSH    BC         ;BC retten
  0024 0604           LD      B,04H      ;DCB-Typ: Ein- und Ausgabe
  0026 181E           JR      0046H      ;und DCB aufrufen
```

```
0028 C30C40            JP     400CH         :RST 28H: Sprung nach 400CH
                                            :(Break-Vektor, freigehalten für
                                            :DOS

; Hole Byte von Tastatur (AF,DE)
; I: -
; O: A = ASCII-Code der gedrückten Taste oder 00H wenn keine Taste gedrückt

0028 111540            LD     DE,4015H      :DE= Adresse des Tastatur DCBs
002E 18E3              JR     0013H         :weiter bei 0013H


0030 C30F40            JP     400FH         :RST 30H: Sprung nach 400FH
                                            :(freigehalten für DOS)

; Ausgabe eines Bytes auf den Bildschirm (AF,DE)
; I: A = ASCII-Code des darzustellenden Zeichens
; O: -

0033 111D40            LD     DE,401DH      :DE= Adresse des Bildschirm DCBs
0036 18E3              JR     001BH         :weiter bei 001BH


0038 C31240            JP     4012H         :RST 38H: Sprung nach 4012H
                                            :(Interrupt-Vektor, freigehalten
                                            :für DOS)

; Ausgabe eines Bytes auf den Drucker (AF,DE)
; I: A = ASCII-Code des zu druckenden Zeichens
; O: -

003B 112540            LD     DE,4025H      :DE= Adresse des Drucker DCBs
003E 18DB              JR     001BH         :weiter bei 001BH


0040 C3D905            JP     05D9H         :Eingabe einer Zeile
                                            :(siehe 05D9H)

0043 C9                RET                  :--
0044 00                NOP
0045 00                NOP

0046 C3C203            JP     03C2H         :Sprung zum DCB-Aufruf

; Warte auf Tastendruck (AF,DE)
; I: -
; O: A  = ASCII-Code der gedrückten Taste

0049 CD2B00            CALL   002BH         :Hole Byte von Tastatur
004C B7                OR     A             :Taste gedrückt? (A <> 0)
004D C0                RET    NZ            :Ja: Return
004E 18F9              JR     0049H         :Nein: Wiederhole die Routine
```

```
; Decodiertabelle für die Tastaturroutine
; ASCII-Codes der entsprechenden Tasten

  0050 0D                                          :RETURN
  0051 0D                                          :RETURN SHIFT
  0052 1F                                          :CLEAR
  0053 1F                                          :CLEAR SHIFT
  0054 01                                          :BREAK
  0055 01                                          :BREAK SHIFT
  0056 5B                                          :HOCHPFEIL
  0057 1B                                          :HOCHPFEIL SHIFT
  0058 0A                                          :TIEFPFEIL
  0059 1A                                          :TIEFPFEIL SHIFT
  005A 08                                          :LINKSPFEIL
  005B 18                                          :LINKSPFEIL SHIFT
  005C 09                                          :RECHTSPFEIL
  005D 19                                          :RECHTSPFEIL SHIFT
  005E 20                                          :LEERTASTE
  005F 20                                          :LEERTASTE SHIFT

; Warteschleife (AF,BC)
; I: BC = Zähler (Wartezeit = BC * 11.3 us (Mikrosekunden))
; O: -

  0060 0B               DEC     BC                 :BC = BC-1
  0061 78               LD      A,B
  0062 B1               OR      C                  :B und C=0?
  0063 20FB             JR      NZ,0060H           :Nein: weiterzählen
  0065 C9               RET                        :Ja: Fertig

; RESET-Einsprung (nach Drücken der beiden RESET-Tasten)

  0066 01181A           LD      BC,1A18H           :BC = Adresse für Rücksprung ins
                                                   :BASIC
  0069 C3CA05           JP      05CAH              :weiter bei 05CAH

; Start 4 (Fortsetzung von 05C7H)
; RAM auf BASIC vorbereiten

  006C 31F841           LD      SP,41F8H           :Stackpointer setzen
  006F 118040           LD      DE,4080H           :ROM-Bereich von 18F7H bis 191CH
  0072 21F718           LD      HL,18F7H           :ins RAM von 4080H bis 40A6H
  0075 012700           LD      BC,0027H           :verschieben
  0078 C34001           JP      0140H              :weiter bei 0140H

; Start 6 wenn keine ROM-Cassette mit einem Basicprogramm vorhanden ist
; (Fortsetzung von 0149H)

  007B 210158           LD      HL,5801H           :HL = Zeiger auf Anfang des
                                                   :BASIC-Programms
  007E 3A80F8           LD      A,(0F880H)
  0081 CB4F             BIT     01H,A              :MOD SEL gedrückt?
  0083 2009             JR      NZ,008EH           :Ja: weiter bei 008EH
  0085 22A440           LD      (40A4H),HL         :Programmstart Adresse speichern
  0088 CD4638           CALL    3846H              :FCLS
```

```
: Start 6 wenn eine ROM-Cassette mit einem Basicprogramm vorhanden ist
: (Fortsetzung von 014CH)

  008B 22A440      LD      (40A4H),HL    ;Programmstartadresse speichern
  008E 21E541      LD      HL,41E5H      ;HL : Start des Zeilenbuffers-3
                                         ;Start markieren:
  0091 363A        LD      (HL),3AH      ;':'
  0093 23          INC     HL
  0094 70          LD      (HL),B        ;00H
  0095 23          INC     HL
  0096 362C        LD      (HL),2CH      ;','
  0098 23          INC     HL
  0099 22A740      LD      (40A7H),HL    ;Bufferadresse speichern
                                         ;Disk Basic Vektoren sperren
  009C 113B01      LD      DE,013BH      ;DE = Adresse der Fehlerroutine
  009F 061C        LD      B,1CH         ;B = Zähler (28 Vektoren)
  00A1 215241      LD      HL,4152H      ;HL : Vektortabelle im RAM
                                         ;Vektoren aufbauen:
                                         ;28 mal JP 013BH ins RAM setzen
  00A4 36C3        LD      (HL),0C3H     ;C3H = JP
  00A6 23          INC     HL
  00A7 73          LD      (HL),E        ;LSB der Adresse
  00A8 23          INC     HL
  00A9 72          LD      (HL),D        ;MSB der Adresse
  00AA 23          INC     HL
  00AB 10F7        DJNZ    00A4H         ;nächsten Vektor
                                         ;Weitere Vektoren durch einsetzen
                                         ;von C9H (=RET) sperren
  00AD 0615        LD      B,15H         ;21 Vektoren
  00AF 36C9        LD      (HL),0C9H     ;C9H einsetzen
  00B1 23          INC     HL
  00B2 23          INC     HL            ;und 2 Bytes freilassen, damit
  00B3 23          INC     HL            ;der RET-Befehl in einen
                                         ;JP-Befehl geändert werden kann
  00B4 10F9        DJNZ    00AFH         ;nächsten Vektor
  00B6 2AA440      LD      HL,(40A4H)    ;HL : Start des BASIC-Programms
  00B9 2B          DEC     HL            ;HL -1
  00BA 70          LD      (HL),B        ;Start - 1 mit 00H markieren
  00BB CD7038      CALL    3870H         ;CRTC initialisieren
  00BE CD8F1B      CALL    1B8FH         ;Stackpointer setzen
  00C1 CDAF06      CALL    06AFH         ;ROM-Cassette vorhanden ?
  00C4 CDC901      CALL    01C9H         ;CLS
  00C7 211801      LD      HL,0118H      ;HL = Text 'MEM SIZE'
  00CA CDA728      CALL    28A7H         ;Text anzeigen
  00CD CDB31B      CALL    1BB3H         ;'? ' ausgeben und
                                         ;auf Eingabe warten
  00D0 38F5        JR      C,00C7H       ;BREAK gedrückt ?
                                         ;Ja: Eingabe wiederholen
  00D2 D7          RST     10H           ;Erstes Zeichen <> ' ' nach A
  00D3 B7          OR      A             ;Zeichen = 00H ?
                                         ;('RETURN' gedrückt ?)
  00D4 2013        JR      NZ,00E9H      ;Nein: Zahl eingegeben
```

```
00D6 210040        LD     HL,4000H      ;Ja: HL : Start des RAMs
00D9 23            INC    HL            ;HL +1
00DA 7C            LD     A,H           ;HL = 0000 ?
00DB B5            OR     L             ;(gesamten Speicher getestet)
00DC 281C          JR     Z,00FAH       ;Ja: weiter bei 00FAH
00DE 7E            LD     A,(HL)        ;Hole Byte aus Speicher
00DF 47            LD     B,A           ;Byte retten
00E0 2F            CPL                  ;A komplementieren
00E1 77            LD     (HL),A        ;und abspeichern
00E2 BE            CP     (HL)          ;steht es auch im RAM ?
00E3 70            LD     (HL),B        ;alten Wert zurück
00E4 28F3          JR     Z,00D9H       ;Ja: nächstes Byte
                                        ;Nein: HL = Endadresse des
                                        ;Speichers + 1
00E6 25            DEC    H             ;256 Bytes abziehen für
                                        ;SHAPE-Tabelle
00E7 1811          JR     00FAH         ;weiter bei 00FAH
```

Zahl bei MEM SIZE-Frage eingegeben

```
00E9 CD5A1E        CALL   1E5AH         ;Eingegebene Zahl auswerten
00EC B7            OR     A             ;wurden nur Ziffern eingegeben ?
00ED C29719        JP     NZ,1997H      ;Nein: SN-Error
00F0 EB            EX     DE,HL
00F1 2B            DEC    HL            ;HL = MEM SIZE - 1
00F2 3E8F          LD     A,8FH         ;A = Testbyte
00F4 46            LD     B,(HL)        ;alten Wert retten
00F5 77            LD     (HL),A        ;Testbyte nach (HL) schreiben
00F6 BE            CP     (HL)          ;Steht es auch dort ?
00F7 70            LD     (HL),B        ;alten Wert zurück
00F8 20CD          JR     NZ,00C7H      ;Nein: MEM SIZE neu abfragen
```

Speichergrenzen festlegen:
HL = Adresse des höchsten verfügbaren Speicherplatzes + 1

```
00FA 2B            DEC    HL            ;HL -1
00FB 111444        LD     DE,4414H      ;DE = 4414H
00FE DF            RST    18H           ;HL und DE vergleichen
00FF DA7A19        JP     C,197AH       ;OM-Error wenn MEM SIZE < 4414H
0102 11CEFF        LD     DE,0FFCEH     ;DE = -50
0105 22B140        LD     (40B1H),HL    ;TOPMEM abspeichern
0108 19            ADD    HL,DE         ;HL = HL + (-50): CLEAR 50
0109 22A040        LD     (40A0H),HL    ;Anfangsadresse des
                                        ;Stringspeichers retten
010C CD4D1B        CALL   1B4DH         ;NEW und CLEAR ausführen
010F 212101        LD     HL,0121H      ;HL : Text 'COLOUR BASIC'
0112 CDA728        CALL   28A7H         ;Text ausgeben
0115 C3191A        JP     1A19H         ;und ins BASIC springen
```

```
; Texte:

  0118 4D                                              ;'MEM SIZE'
  0119 45
  011A 4D
  011B 20
  011C 53
  011D 49
  011E 5A
  011F 45
  0120 00

  0121 43                                              ;'COLOUR BASIC'
  0122 4F
  0123 4C
  0124 4F
  0125 55
  0126 52
  0127 20
  0128 42
  0129 41
  012A 53
  012B 49
  012C 43
  012D 0D
  012E 00

  012F FF                    RST       38H             ;--
  0130 FF                    RST       38H
  0131 FF                    RST       38H

; X = CHECK ( Bitnummer , Adresse )

  0132 C36B01                JP        016BH           ;weiter bei 016BH

; SET Bitnummer , Adresse

  0135 C34F01                JP        014FH           ;weiter bei 014FH

; RESET Bitnummer , Adresse

  0138 C35D01                JP        015DH           ;weiter bei 015DH

; Einsprung bei gesperrten Disk Basic Vektoren

  013B 1E2C                  LD        E,2CH           ;E = Fehlercode für SN-Error ohne
                                                       ;EDIT-Aufruf
  013D C3A219                JP        19A2H           ;und zur Errorroutine springen
```

```
; Start 5 (Fortsetzung von 0078H)

  0140 EDB0              LDIR                    :Verschiebung ausführen
  0142 2101C0            LD      HL,0C001H       :HL = Zeiger auf ROM-Cassette
  0145 3A00C0            LD      A,(0C000H)      :ROM-Cassette mit Basicprogramm
                                                 :vorhanden ?
  0148 B7                OR      A
  0149 C27B00            JP      NZ,007BH        :Nein: weiter bei 007BH
  014C C38B00            JP      008BH           :Ja: weiter bei 008BH

; Fortsetzung der SET-Routine von 0135H

  014F CD8301            CALL    0183H           :Bitnr. und Adresse holen
  0152 3E01              LD      A,01H           :A = Maske für Bit setzen
  0154 07                RLCA                    :verschiebe A bis gewünschtes
  0155 10FD              DJNZ    0154H           :Bit in A gesetzt
  0157 0F                RRCA                    :ein Bit zurück,
                                                 :da B = Bitnr. + 1 war
  0158 47                LD      B,A             :Maske nach B
  0159 1A                LD      A,(DE)          :Byte aus Speicher holen
  015A B0                OR      B               :Bit einblenden
  015B 12                LD      (DE),A          :und zurückschreiben
  015C C9                RET

; Fortsetzung der RESET-Routine von 0138H

  015D CD8301            CALL    0183H           :Bitnr. und Adresse holen
  0160 3EFE              LD      A,0FEH          :A = Maske für Bit löschen
  0162 07                RLCA                    :verschiebe A bis gewünschtes
  0163 10FD              DJNZ    0162H           :Bit in A rückgesetzt
  0165 0F                RRCA                    :ein Bit zurück,
                                                 :da B = Bitnr. + 1 war
  0166 47                LD      B,A             :Maske nach B
  0167 1A                LD      A,(DE)          :Byte aus Speicher holen
  0168 A0                AND     B               :Bit ausblenden
  0169 12                LD      (DE),A          :und zurückschreiben
  016A C9                RET

; Fortsetzung der CHECK-Routine von 0132H

  016B D7                RST     10H             :Zeichen nach 'CHECK' in A
  016C CF                RST     08H             :Klammer auf ?
  016D 28                DEFB    '('             :sonst SN-Error
  016E CD8301            CALL    0183H           :Bitnr. und Adresse holen
  0171 E5                PUSH    HL              :PTZ retten
  0172 1A                LD      A,(DE)          :Byte aus Speicher holen
  0173 1F                RRA                     :und gewünschtes Bit ins
  0174 10FD              DJNZ    0173H           :Carry-Flag setzen
  0176 21FFFF            LD      HL,0FFFFH       :HL = -1 (für Bit gesetzt)
  0179 3801              JR      C,017CH         :Sprung wenn Bit gesetzt
  017B 23                INC     HL              :sonst HL = 0 (Bit gelöscht)
  017C CD9A0A            CALL    0A9AH           :HL als INT nach X schreiben
  017F E1                POP     HL              :PTZ zurück
  0180 CF                RST     08H             :Klammer geschlossen ?
  0181 29                DEFB    ')'
  0182 C9                RET
```

```
: UPRO für SET, RESET und CHECK
: Bitnummer und Adresse aus Programmtext holen
: I: HL = PTZ
: O: B  = Bitnummer + 1
:    DE = Adresse

  0183 CD1C2B         CALL    2B1CH        :Bitnummer nach A
  0186 FE08           CP      08H          :grösser als 7 ?
  0188 D24A1E         JP      NC,1E4AH     :Ja: FC-Error
  018B F5             PUSH    AF           :Bitnr. retten
  018C CF             RST     08H          :Beide Zahlen durch Komma
  018D 2C             DEFB    ','          :getrennt ?
  018E CD022B         CALL    2B02H        :Adresse nach DE
  0191 F1             POP     AF           :Bitnr. zurück
  0192 47             LD      B,A
  0193 04             INC     B            :B = Bitnr. + 1
  0194 C9             RET

  0195 FF             RST     38H          :--
  0196 FF             RST     38H
  0197 FF             RST     38H
  0198 FF             RST     38H
  0199 FF             RST     38H
  019A FF             RST     38H
  019B FF             RST     38H
  019C FF             RST     38H

: X = INKEY$

  019D D7             RST     10H          :PTZ auf nächstes Zeichen
  019E E5             PUSH    HL           :und retten
  019F 3A9940         LD      A,(4099H)    :ist schon vorher eine Taste
  01A2 B7             OR      A            :gedrückt worden ?
  01A3 2006           JR      NZ,01ABH     :Ja: Wert übergeben und zurück
  01A5 CD5803         CALL    0358H        :Nein: Jetzt Taste gedrückt ?
  01A8 B7             OR      A
  01A9 2811           JR      Z,01BCH      :Nein: Nullstring als Ergebnis
  01AB F5             PUSH    AF           :Tastencode retten
  01AC AF             XOR     A            :Letzten Tastencode löschen
  01AD 329940         LD      (4099H),A
  01B0 3C             INC     A            :Ein Byte im Stringspeicher
  01B1 CD5728         CALL    2857H        :reservieren
  01B4 F1             POP     AF           :Tastencode zurück
  01B5 2AD440         LD      HL,(40D4H)   :HL : Speicherplatz für neuen
                                           :String
  01B8 77             LD      (HL),A       :Zeichen abspeichern
  01B9 C38428         JP      2884H        :und VT auf STR setzen
```

```
; Keine Taste gedrückt
; Nullstring als Ergebnis übergeben

  01BC 212819            LD      HL,1928H       ;HL : Nullstring
  01BF 222141            LD      (4121H),HL     ;als Ergebnis setzen
  01C2 3E03              LD      A,03H          ;VT auf STR
  01C4 32AF40            LD      (40AFH),A
  01C7 E1                POP     HL             ;PTZ zurück
  01C8 C9                RET

: CLS

  01C9 3E1C              LD      A,1CH          ;Cursor HOME
  01CB CD3A03            CALL    033AH
  01CE 3E1F              LD      A,1FH          ;Bildschirm bis zum Ende löschen
  01D0 C33A03            JP      033AH

: RANDOM

  01D3 ED5F              LD      A,R            ;Refresh-Register nach A
  01D5 32AB40            LD      (40ABH),A      ;und im RAM ablegen
  01D8 C9                RET

: Pegelwechsel an Cassettenport ausgeben

  01D9 3A1C43            LD      A,(431CH)      ;A = letzter Wert
  01DC EE01              XOR     01H            ;Bit 0 wechseln
  01DE D3FF              OUT     (0FFH),A       ;und ausgeben
  01E0 321C43            LD      (431CH),A      ;neuen Wert zurückschreiben
  01E3 C9                RET

: * blinken

  01E4 3A2744            LD      A,(4427H)      ;'*' oder ' ' aus Bildschirm-
                                                ;speicher holen
  01E7 EE0A              XOR     0AH            ;'*' und ' ' vertauschen
  01E9 322744            LD      (4427H),A      ;neues Zeichen zurückschreiben
  01EC C9                RET

: Ein Byte von Cassette lesen (AF)
: I: -
: O: A = gelesenes Byte

  01ED D9                EXX                    ;Registersatz vertauschen
  01EE 0608              LD      B,08H          ;8 Bits werden gelesen
  01F0 1600              LD      D,00H          ;Byte auf 00H setzen
  01F2 CDFA01            CALL    01FAH          ;Ein Bit lesen und in D schieben
  01F5 10FB              DJNZ    01F2H          ;nächstes Bit
  01F7 7A                LD      A,D            ;komplettes Byte in A
  01F8 D9                EXX                    ;alte Register wiederherstellen
  01F9 C9                RET
```

```
; Ein Bit von Cassette lesen und in D schieben

  01FA C5               PUSH    BC          :BC (Zähler) retten
                                            ;Auf Zeitimpuls warten:
  01FB DBFF             IN      A,(0FFH)    ;Port FFH abfragen
  01FD E601             AND     01H         ;Pegel ausmaskieren
  01FF 5F               LD      E,A         ;und in E speichern
  0200 DBFF             IN      A,(0FFH)    ;Port FFH abfragen
  0202 E601             AND     01H         ;Pegel ausmaskieren
  0204 AB               XOR     E           ;und mit letztem Wert vergleichen
  0205 1F               RRA                 ;Ergebnis ins Carry-Flag schieben
  0206 30F8             JR      NC,0200H    ;und auf Pegeländerung warten
                                            ;(Clock-Impuls suchen)
  0208 3C               INC     A           ;A = 1 (wegen AND 01H und RRA)
  0209 AB               XOR     E           ;A = neuer Pegelwert
  020A 5F               LD      E,A         ;in E abspeichern für Vergleich
  020B 3A1243           LD      A,(4312H)   ;A = Wert für Zeitschleife
  020E 47               LD      B,A         ;nach B
  020F 10FE             DJNZ    020FH       ;und B runterzählen
  0211 DBFF             IN      A,(0FFH)    ;Jetzt neuen Pegelwert holen
  0213 E601             AND     01H
  0215 AB               XOR     E           ;und mit altem Wert vergleichen
                                            ;A ist 1 wenn ein Pegelwechsel
                                            ;erkannt wurde
  0216 CB22             SLA     D           ;D verschieben
  0218 B2               OR      D           ;neuen Wert einblenden
  0219 57               LD      D,A         ;und in D abspeichern
  021A C1               POP     BC          ;Zähler züruckholen
  021B C9               RET

; Zweimal dasselbe Byte auf Cassette schreiben

  021C CD1F02           CALL    021FH       ;Erstes Byte schreiben

; Ein Byte auf Cassette schreiben (AF)
; I: A = zu schreibendes Byte
; O: A = geschriebenes Byte

  021F D9               EXX                 ;Register retten
  0220 F5               PUSH    AF          ;Byte retten
  0221 0E08             LD      C,08H       ;8 Bits schreiben
  0223 57               LD      D,A         ;D = Byte
  0224 CDD901           CALL    01D9H       ;Clockimpuls ausgeben
  0227 3A1043           LD      A,(4310H)   ;Erste Warteschleife ausführen
  022A 47               LD      B,A
  022B 10FE             DJNZ    022BH
  022D 7A               LD      A,D         ;A = Byte
  022E 07               RLCA                ;nächstes Bit nach Carry schieben
  022F 57               LD      D,A         ;und verschobenen Wert nach D
  0230 DCD901           CALL    C,01D9H     ;Pegelwechsel ausgeben wenn
                                            ;Bit = 1
```

```
0233 3A1143          LD      A,(4311H)     :zweite Warteschleife
0236 47              LD      B,A
0237 10FE            DJNZ    0237H
0239 0D              DEC     C             :Bitzähler runterzählen
023A 20E8            JR      NZ,0224H      :nächstes Bit ausgeben
023C F1              POP     AF            :Byte zurück nach A
023D D9              EXX                   :Register wiederherstellen
023E C9              RET

; Leader und Sync auf Cassette schreiben

023F 06FF            LD      B,0FFH        :255 mal
0241 3EAA            LD      A,0AAH        :das Byte AAH
0243 CD1F02          CALL    021FH         :auf Cassette schreiben
0246 10FB            DJNZ    0243H
0248 3E66            LD      A,66H         :Sync = 66H
024A 18D3            JR      021FH         :auf Cassette schreiben

; Leader und Sync suchen

024C E5              PUSH    HL            :Register retten
024D D5              PUSH    DE            :(mit PUSH, da EXX bei 01EDH
024E C5              PUSH    BC            :gebraucht wird)
024F 216935          LD      HL,3569H      :HL : Farbcode Tabelle
0252 110000          LD      DE,0000H      :DE = 0000H
0255 3A2340          LD      A,(4023H)     :A = Jetziger Farbcode
0258 5F              LD      E,A           :DE = Offset zum Tabellenwert
0259 19              ADD     HL,DE         :HL = Zeiger zum Farbcode
025A 7E              LD      A,(HL)        :A = Farbcode aus Tabelle
025B 3226F0          LD      (0F026H),A    :für beide Sterne
025E 3227F0          LD      (0F027H),A    :im Farbspeicher ablegen
0261 01AA80          LD      BC,80AAH      :B = Zähler, C = gesuchtes Byte
0264 CDFA01          CALL    01FAH         :Ein Bit von Cassette lesen,
                                           :in D schieben und nach A laden
0267 B9              CP      C             :Byte gefunden ?
0268 20F7            JR      NZ,0261H      :Nein: Suche neu beginnen
026A 3EFF            LD      A,0FFH        :Ja: Alle Bits in C umkehren
026C A9              XOR     C             :also jetzt nach 55H suchen
026D 4F              LD      C,A
026E 10F4            DJNZ    0264H         :weitersuchen bis 128 mal AAH
                                           :gefunden wurde
0270 CDFA01          CALL    01FAH         :Jetzt Sync (66H) suchen
0273 FE66            CP      66H           :gefunden ?
0275 20F9            JR      NZ,0270H      :Nein: weitersuchen
0277 3E2A            LD      A,2AH         :Ja: Zwei Sterne in rechter
0279 322644          LD      (4426H),A     :Bildschirmecke
027C 322744          LD      (4427H),A     :anzeigen
027F C1              POP     BC            :Register zurück
0280 D1              POP     DE
0281 E1              POP     HL
0282 C9              RET
```

```
0283 FF          RST     38H        ;--
0284 FF          RST     38H
0285 FF          RST     38H
0286 FF          RST     38H
0287 FF          RST     38H
0288 FF          RST     38H
0289 FF          RST     38H
028A FF          RST     38H
028B FF          RST     38H
028C FF          RST     38H
028D FF          RST     38H
028E FF          RST     38H
028F FF          RST     38H
0290 FF          RST     38H
0291 FF          RST     38H
0292 FF          RST     38H
0293 FF          RST     38H
0294 FF          RST     38H
0295 FF          RST     38H
0296 FF          RST     38H
0297 FF          RST     38H
0298 FF          RST     38H
0299 FF          RST     38H
029A FF          RST     38H
029B FF          RST     38H
029C FF          RST     38H
029D FF          RST     38H
029E FF          RST     38H
029F FF          RST     38H
02A0 FF          RST     38H
02A1 FF          RST     38H
02A2 FF          RST     38H
02A3 FF          RST     38H
02A4 FF          RST     38H
02A5 FF          RST     38H
02A6 FF          RST     38H
02A7 FF          RST     38H
02A8 FF          RST     38H
02A9 FF          RST     38H
02AA FF          RST     38H
02AB FF          RST     38H
```

```
; Startadresse des geladenen Programms vom Band lesen (für SYSTEM)

  02AC CD1403        CALL   0314H       :2 Bytes (Startadresse) lesen
  02AF 22DF40        LD     (40DFH),HL  ;und abspeichern

; SYSTEM

  02B2 CDE241        CALL   41E2H       ;DOS
  02B5 318842        LD     SP,4288H    :Stack in den Tastaturbuffer
                                        :legen, so daß er nicht stört
  02B8 CDFE20        CALL   20FEH       :Neue Zeile beginnen
  02BB 3E2A          LD     A,2AH       :Stern '*' ausgeben
  02BD CD2A03        CALL   032AH
  02C0 CDB31B        CALL   1BB3H       :und nach Filename fragen
  02C3 DA6600        JP     C,0066H     :Zurück ins BASIC wenn BREAK
                                        :gedrückt wurde
  02C6 D7            RST    10H         :erstes Zeichen holen
  02C7 CA9719        JP     Z,1997H     :SN-Error wenn nichts eingegeben
  02CA FE2F          CP     2FH         :ist es '/' ?
  02CC 284F          JR     Z,031DH     :Ja: weiter bei 031DH
  02CE CD4C02        CALL   024CH       :Nein: Leader und Sync suchen
  02D1 CDED01        CALL   01EDH       :erstes Byte von Band holen
  02D4 FE55          CP     55H         :Ist es ein SYSTEM-Programm ?
  02D6 20F9          JR     NZ,02D1H    :Nein: weitersuchen
  02D8 0606          LD     B,06H       :6 Zeichen für Filename
  02DA 7E            LD     A,(HL)      :Zeichen aus Buffer holen
  02DB B7            OR     A           :Zeilenende gefunden ?
  02DC 2809          JR     Z,02E7H     :Ja: Programm laden
  02DE CDED01        CALL   01EDH       :Nein: Filename laden
  02E1 BE            CP     (HL)        :und mit gesuchtem Namen
                                        :vergleichen
  02E2 20ED          JR     NZ,02D1H    :Gesuchtes File gefunden ?
                                        :Nein: Nächstes File suchen
  02E4 23            INC    HL          :Ja: Zeiger +1
  02E5 10F3          DJNZ   02DAH       :und nächstes Zeichen vergleichen
  02E7 CDE401        CALL   01E4H       :Rechten Stern blinken lassen
  02EA CDED01        CALL   01EDH       :Blockmarkierung von Band lesen
  02ED FE78          CP     78H         :Ende gefunden ?
  02EF 28BB          JR     Z,02ACH     :Ja: Startadresse holen und
                                        :wieder '*? ' ausgeben
  02F1 FE3C          CP     3CH         :Blockanfang gefunden
  02F3 20F5          JR     NZ,02EAH    :Nein: Neues Byte suchen
  02F5 CDED01        CALL   01EDH       :Blocklänge nach B
  02F8 47            LD     B,A         :B = Anzahl der zu lesenden Bytes
  02F9 CD1403        CALL   0314H       :und Adresse nach HL
  02FC 85            ADD    A,L         :Prüfsumme errechnen
  02FD 4F            LD     C,A         :und in C speichern
```

```
  02FE CDED01           CALL    01EDH        :Byte lesen
  0301 77               LD      (HL),A       :und abspeichern
  0302 23               INC     HL           :Zeiger +1
  0303 81               ADD     A,C          :Prüfsumme addieren
  0304 4F               LD      C,A          :und nach C laden
  0305 10F7             DJNZ    02FEH        :nächstes Byte des Blocks laden
  0307 CDED01           CALL    01EDH        :Block zu Ende: Prüfsumme holen
  030A B9               CP      C            :und mit C vergleichen
  030B 28DA             JR      Z,02E7H      :Prüsumme identisch ?
                                             :Ja: nächsten Block laden
  030D 3E43             LD      A,43H        :Nein: A = 'C'
  030F 322644           LD      (4426H),A    :Prüfsummenfehler anzeigen
  0312 18D6             JR      02EAH        :aber trotzdem weiterladen

; UPRO für SYSTEM:
; Adresse (Blockadresse oder Startadresse) vom Band holen
; I: -
; O: HL = gelesene Adresse

  0314 CDED01           CALL    01EDH        :Adresse vom Band lesen
  0317 6F               LD      L,A          :LSB nach L
  0318 CDED01           CALL    01EDH
  031B 67               LD      H,A          :MSB nach H
  031C C9               RET

; '/' bei SYSTEM eingegeben

  031D EB               EX      DE,HL        :DE = Textzeiger
  031E 2ADF40           LD      HL,(40DFH)   :Startadresse nach
  0321 EB               EX      DE,HL        :DE laden
  0322 D7               RST     10H          :wurde eine Startadresse
                                             :angegeben ?
  0323 C45A1E           CALL    NZ,1E5AH     :Ja: Neue Adresse nach DE
  0326 208A             JR      NZ,02B2H     :SYSTEM neu beginnen wenn
                                             :Blödsinn eingegeben wurde
  0328 EB               EX      DE,HL        :sonst Adresse nach HL laden
  0329 E9               JP      (HL)         :und Ausführung beginnen

; Ausgabe von A auf Bildschirm, Drucker oder Cassette (AF,HL)
; I: A = ASCII-Code des auszugebenden Zeichens
;    (409CH) = Ausgabeflag: 00H = Bildschirm, 01H = Drucker, 80H = Cassette
; O: -

  032A C5               PUSH    BC           :BC retten
  032B 4F               LD      C,A          :Zeichen nach C
  032C CDC141           CALL    41C1H        :DOS
  032F 3A9C40           LD      A,(409CH)    :Flag testen
  0332 B7               OR      A
  0333 79               LD      A,C          :Zeichen zurück nach A
  0334 C1               POP     BC           :BC zurück
  0335 C36405           JP      0564H        :weiter bei 0564H
```

```
0338 FF              RST    38H          ;--
0339 FF              RST    38H

; Ausgabe von A auf den Bildschirm und POS (40A6H) erhöhen

033A D9              EXX                 ;Register retten
033B F5              PUSH   AF           ;Zeichen retten
033C CD3300          CALL   0033H        ;Zeichen ausgeben
033F CD4803          CALL   0348H        ;Neuen POS errechnen
0342 32A640          LD     (40A6H),A    ;und abspeichern
0345 F1              POP    AF           ;Zeichen zurück
0346 D9              EXX                 ;Register zurück
0347 C9              RET

; Neuen POS errechnen (AF,DE)
; I: -
; O: A = neuer POS-Wert

0348 E5              PUSH   HL           ;PTZ retten
0349 2A2040          LD     HL,(4020H)   ;HL = Cursoradresse
034C 110044          LD     DE,4400H     ;DE : Bildschirmanfang
034F B7              OR     A            ;Carryflag auf 0 setzen
0350 C3D904          JP     04D9H        ;weiter bei 04D9H

0353 FF              RST    38H          ;--
0354 FF              RST    38H

; Neuen POS errechen (wie 0348H)

0355 C39D30          JP     309DH        ;weiter bei 309DH

; Tastaturabfrage (AF)
; (wie 002BH aber mit DOS und DE Rettung)

0358 CDC441          CALL   41C4H        ;DOS
035B D5              PUSH   DE           ;DE retten
035C CD2B00          CALL   002BH        ;Tastaturabfrage
035F D1              POP    DE           ;DE zurück
0360 C9              RET

; Eingabe einer Zeile mit max. 240 Zeichen in den Zeilenbuffer (AF,DE,HL)
; I: -
; O: Carryflag = 1 wenn Break gedrückt wurde

0361 AF              XOR    A            ;A = 00H
0362 329940          LD     (4099H),A    ;Letzten Tastencode löschen
0365 32A640          LD     (40A6H),A    ;POS auf 0 setzen
0368 CDAF41          CALL   41AFH        ;DOS
036B C5              PUSH   BC           ;BC retten
036C 2AA740          LD     HL,(40A7H)   ;HL : Zeilenbuffer
036F 06F0            LD     B,0F0H       ;B = Maximale Zeichenzahl (240)
0371 CDD905          CALL   05D9H        ;Zeile eingeben lassen
0374 F5              PUSH   AF           ;Flags retten
0375 48              LD     C,B          ;C = Anzahl der eingegebenen
                                         ;Zeichen
```

```
0376 0600            LD     B,00H         ;B = 0 -> BC = Länge der Zeile
0378 09              ADD    HL,BC         ;HL zeigt auf letztes Zeichen + 1
0379 3600            LD     (HL),00H      ;Zeile mit 00H abschließen
037B 2AA740          LD     HL,(40A7H)    ;HL : Anfang des Zeilenbuffers
037E F1              POP    AF            ;Flags zurück
037F C1              POP    BC            ;BC zurück
0380 2B              DEC    HL            ;HL = HL - 1 (für RST 10H)
0381 D8              RET    C             ;CY = 1 wenn Break gedrückt
0382 AF              XOR    A             ;CY = 0
0383 C9              RET

; Warten auf Tastendruck (AF)
; (wie 0049H aber mit DOS und DE Rettung)

0384 CD5803          CALL   0358H         ;Taste holen
0387 B7              OR     A             ;Neue Taste gedrückt ?
0388 C0              RET    NZ            ;Ja: Rücksprung
0389 18F9            JR     0384H         ;Nein: Auf Tastendruck warten

; Ausgaben auf Drucker abschließen (AF)
; I: -
; O: DPOS = 0

038B AF              XOR    A             ;Nächste Ausgabe auf
038C 329C40          LD     (409CH),A     ;Bildschirm bringen
038F 3A9B40          LD     A,(409BH)     ;DPOS = 0 ?
0392 B7              OR     A
0393 C8              RET    Z             ;Ja: Fertig
0394 3E0D            LD     A,0DH         ;Nein: CR zum Drucker
0396 D5              PUSH   DE            ;DE retten
0397 CD9C03          CALL   039CH         ;CR ausgeben (Drucker auf neue
                                          ;Zeile bringen)
039A D1              POP    DE            ;DE zurück
039B C9              RET

; Ein Zeichen auf dem Drucker ausdrucken und Steuerzeichen abfangen

039C F5              PUSH   AF            ;Register retten
039D D5              PUSH   DE
039E C5              PUSH   BC
039F 4F              LD     C,A           ;Zeichen in C retten
03A0 1E00            LD     E,00H         ;E = 00H für evtl. neue DPOS
03A2 FE0C            CP     0CH           ;Ist es Form Feed ?
03A4 2810            JR     Z,03B6H       ;Ja: DPOS auf 0 und Zeichen
                                          ;ausdrucken
03A6 FE0A            CP     0AH           ;Ist es Line Feed ?
03A8 2003            JR     NZ,03ADH      ;Nein: Weiterprüfen
03AA 3E0D            LD     A,0DH         ;Ja: Umwandeln in ein
                                          ;Carriage Return
03AC 4F              LD     C,A           ;und in C ablegen
```

```
03AD FE0D        CP     0DH          ;Ist es Carriage Return ?
03AF 2805        JR     Z,03B6H      ;Ja: DPOS auf 0 und Zeichen
                                     ;ausdrucken
03B1 3A9B40      LD     A,(409BH)    ;Nein: DPOS holen
03B4 3C          INC    A            ;um eins erhöhen
03B5 5F          LD     E,A          ;und in E ablegen
03B6 7B          LD     A,E          ;Neuen DPOS abspeichern
03B7 329B40      LD     (409BH),A
03BA 79          LD     A,C          ;Zeichen nach A
03BB CD3B00      CALL   003BH        ;und ausdrucken
03BE C1          POP    BC           ;Register zurück
03BF D1          POP    DE
03C0 F1          POP    AF
03C1 C9          RET

; DCB-Aufruf (AF) (siehe 0046H)
; I: A  = Zeichen (wenn Ausgabe)
;    B  = DCB-Typ
;    DE = DCB-Adresse
; O: A = Zeichen (wenn Eingabe)

03C2 E5          PUSH   HL           ;Register retten
03C3 DDE5        PUSH   IX
03C5 D5          PUSH   DE
03C6 DDE1        POP    IX           ;IX = DCB-Adresse
03C8 D5          PUSH   DE           ;Stack wieder richten
03C9 21DD03      LD     HL,03DDH     ;Returnadresse im Stack
03CC E5          PUSH   HL           ;ablegen
03CD 4F          LD     C,A          ;Zeichen in C retten
03CE 1A          LD     A,(DE)       ;DCB-Typ von DCB holen
03CF A0          AND    B            ;nicht benötigte Bits ausmaskiern
03D0 B8          CP     B            ;und mit angegebenem Typ
                                     ;vergleichen
03D1 C23340      JP     NZ,4033H     ;Typ falsch: A auf 00H setzen
                                     ;und zurück ins Programm
                                     ;(4033H wird im DOS verwendet)
03D4 FE02        CP     02H          ;DCB-Typ = Ausgabe ?
03D6 DD6E01      LD     L,(IX+01H)   ;Adresse der DCB-Routine
03D9 DD6602      LD     H,(IX+02H)   ;nach HL laden
03DC E9          JP     (HL)         ;und Routine starten

; Ende einer DCB-Routine

03DD D1          POP    DE           ;Register zurück
03DE DDE1        POP    IX
03E0 E1          POP    HL
03E1 C1          POP    BC
03E2 C9          RET

; Tastaturroutine (wird über DCB-Aufruf angesprochen)

03E3 CDAF06      CALL   06AFH        ;Ist ROM-Cassette vorhanden ?
                                     ;Ja: ROM-Programm starten
03E6 3A80F8      LD     A,(0F880H)   ;CTRL und MOD SEL gedrückt ?
03E9 FE12        CP     12H
03EB 200D        JR     NZ,03FAH     ;Nein: weiter bei 03FAH
03ED CDA938      CALL   38A9H        ;Ja: FGR ausführen
03F0 3A40F8      LD     A,(0F840H)   ;Break gedrückt ?
03F3 CB57        BIT    02H,A
03F5 28F9        JR     Z,03F0H      ;Nein: Auf Break warten
03F7 CDB038      CALL   38B0H        ;Ja: LGR ausführen
03FA 213640      LD     HL,4036H     ;HL : Zwischenspeicher
```

```
03FD 0101F8        LD     BC,0F801H     ;BC : Erste Tastaturzeile
0400 1600          LD     D,00H         ;Zeilenzähler auf 0 setzen
0402 0A            LD     A,(BC)        ;A = Wert der Tastaturzeile
0403 5F            LD     E,A           ;Wert ablegen
0404 AE            XOR    (HL)          ;nur die Bits der Tasten auf 1.
                                        ;die sich seid der letzten
                                        ;Abfrage änderten
0405 73            LD     (HL),E        ;Neuen Zeilencode für nächsten
                                        ;Aufruf abspeichern
0406 A3            AND    E             ;Nur das Bit der Taste auf 1.
                                        ;die seid der letzten Abfrage
                                        ;neu gedrückt wurde
0407 2010          JR     NZ,0419H      ;Wurde eine Taste neu gedrückt ?
                                        ;Ja: ASCII-Code der Taste
                                        ;ermitteln
0409 14            INC    D             ;Nein: Zeilenzähler +1
040A 2C            INC    L             ;Bufferzeiger +1
040B CB01          RLC    C             ;BC auf nächste Zeile erhöhen
040D 30F3          JR     NC,0402H      ;War das schon die letzte Zeile ?
                                        ;Nein: nächste Zeile prüfen
040F 3A80F8        LD     A,(0F880H)    ;Ja: Ist RPT gedrückt ?
0412 CB5F          BIT    03H,A
0414 C2D404        JP     NZ,04D4H      ;Ja: weiter bei 04D4H
0417 AF            XOR    A             ;Nein: Dann A auf 00H setzen
0418 C9            RET                  ;und zurück ins Programm

0419 5F            LD     E,A           ;Bit-Wert der Taste nach E
041A 211840        LD     HL,4018H      ;HL : Flagbyte
041D 3A80F8        LD     A,(0F880H)    ;Ist MOD SEL gedrückt ?
0420 CB4F          BIT    01H,A
0422 C2C904        JP     NZ,04C9H      ;Ja: weiter bei 04C9H
0425 CB67          BIT    04H,A         ;Nein: Wurde CTRL gedrückt ?
0427 C2D004        JP     NZ,04D0H      ;Ja: weiter bei 04D0H
042A 3E07          LD     A,07H         ;War die Taste in der letzten
                                        ;Tastaturzeile und wurde noch
                                        ;nicht abgefangen ?
042C BA            CP     D             ;(Ist der Zeilenzähler = 7 ?)
042D 28E8          JR     Z,0417H       ;Ja: Die Taste war nur 'SHIFT'
                                        ;(der Tastencode ist 00H)
```

```
042F 7A          LD     A,D          ;multipliziere D mit 8
0430 07          RLCA                ;* 2
0431 07          RLCA                ;* 2
0432 07          RLCA                ;* 2
0433 57          LD     D,A          ;da 8 Tasten pro Zeile vorhanden
                                     ;sind
0434 0E01        LD     C,01H        ;C = Maske für folgende Be-
                                     ;rechnung
0436 79          LD     A,C          ;A = Maske
0437 A3          AND    E            ;Tastencode maskieren
0438 2005        JR     NZ,043FH     ;Taste neu gedrückt ?
                                     ;Ja: weiter bei 043FH
043A 14          INC    D            ;Nein: D erhöhen, um den Wert der
                                     ;Taste in der Tastaturmatrix zu
                                     ;ermitteln
043B CB01        RLC    C            ;Maske verschieben für
043D 18F7        JR     0436H        ;nächsten Test

: D gibt jetzt den Wert der Taste in der Tastaturmatrix an
: '§' hat den Wert 00H, 'A' den Wert 01H usw. bis zur Leertaste mit 37H

043F 3A80F8      LD     A,(0F880H)   ;Wurde Shift dazu gedrückt ?
0442 47          LD     B,A          ;Wert in B zwischenspeichern
0443 7A          LD     A,D          ;A = Matrixwert
0444 C640        ADD    A,40H        ;Matrixwert + 40H ergibt
                                     ;korrekten ASCII-Code für die
                                     ;Tasten '§' bis 'F4' (entspricht
                                     ;dem ASCII-Wert 40H bis 5FH)
0446 FE60        CP     60H          ;Ist der Wert in diesem Bereich ?
0448 3013        JR     NC,045DH     ;Nein: weiter bei 045DH
044A CB08        RRC    B            ;Wurde Shift gedrückt ?
                                     ;(Ist Bit 0 von F880H = 1 ?)
044C 3031        JR     NC,047FH     ;Nein: Tastencode in Ordnung
044E C620        ADD    A,20H        ;Ja: 20H aufaddieren für
                                     ;Kleinbuchstaben
0450 57          LD     D,A          ;Wert ablegen
0451 3A40F8      LD     A,(0F840H)   ;Wurde Shift und Tiefpfeil
0454 E610        AND    10H          ;gedrückt ?
0456 7A          LD     A,D          ;Wert zurück
0457 2826        JR     Z,047FH      ;Nein: Wert in Ordnung
0459 D660        SUB    60H          ;Ja: Wert in Controlcode
                                     ;umwandeln
045B 1822        JR     047FH        ;und damit weiterarbeiten
```

```
: Es wurden Zahlen, Sonderzeichen oder Kontrolltasten gedrückt:
: A hat einen Wert zwischen 60H (für '0') und 77H (für Leertaste)

 045D D670         SUB    70H          ;Wurde Kontrolltaste gedrückt ?
 045F 3010         JR     NC,0471H     ;Ja: Wert der Tabelle ab 0050H
                                       ;entnehmen
 0461 C640         ADD    A,40H        ;Wert berichtigen
 0463 FE3C         CP     3CH          ;Wurde eine der Tasten
                                       ;',' '-' '.' oder '/' gedrückt ?
 0465 3802         JR     C,0469H      ;Nein: Der richtige ASCII-Wert
                                       ;steht in A
 0467 EE10         XOR    10H          ;Ja: Bei diesen Tasten sind die
                                       ;Symbole vertauscht, d. h. die
                                       ;Vertauschung muß durch XOR 10H
                                       ;rückgängig gemacht werden
                                       ;(Siehe ASCII-Tabelle)
 0469 CB08         RRC    B            ;Shift gedrückt ?
 046B 3012         JR     NC,047FH     ;Nein: Wert übergeben
 046D EE10         XOR    10H          ;Ja: Werte von 20H bis 2FH mit
                                       ;30H bis 3FH vertauschen
 046F 180E         JR     047FH

: Kontrolltaste gedrückt
: Werte aus der Tabelle ab 0050H entnehmen

 0471 07           RLCA                ;A = A * 2 als Tabellenoffset
 0472 CB08         RRC    B            ;Shift gedrückt ?
 0474 3001         JR     NC,0477H     ;Nein: Offset in Ordnung
 0476 3C           INC    A            ;Ja: Für Shift-Wert, Offset um
                                       ;eins erhöhen (siehe 0050H)
 0477 215000       LD     HL,0050H     ;HL = Tabellenzeiger
 047A 5F           LD     E,A          ;DE = Offset
 047B 1600         LD     D,00H
 047D 19           ADD    HL,DE        ;HL zeigt auf Tastencode
 047E 7E           LD     A,(HL)       ;Code nach A holen

: Der korrekte ASCII-Code steht in A, nun muss noch die Sondertaste
: 'MOD SEL' ausgewertet werden

 047F 211840       LD     HL,4018H     ;HL : Flagbyte
 0482 CB76         BIT    06H,(HL)     ;Ist MOD SEL an ? (Grafikzeichen)
 0484 2824         JR     Z,04AAH      ;Nein: Tastaturroutine beenden
 0486 FE2B         CP     2BH          ;Ist der Wert < 2BH ?
 0488 3820         JR     C,04AAH      ;Ja: Diese Tasten haben kein
                                       ;Grafikzeichen zugeordnet
 048A FE30         CP     30H          ;Ist der Wert zwischen 2BH und
                                       ;2FH ?
 048C 3004         JR     NC,0492H     ;Nein: weiterprüfen
 048E D62B         SUB    2BH          ;Ja: Ziehe 2BH ab
 0490 1816         JR     04A8H        ;und addiere C0H dazu, sodaß
                                       ;'+' den Wert C0H und '/' den
                                       ;Wert C4H bekommt
```

```
0492 FE3B          CP      3BH           :Ist der Wert kleiner als ':' ?
0494 3814          JR      C.04AAH       :Ja: Keine Grafikzeichen
0496 FE5B          CP      5BH           :Ist der Wert größer als 'Z' ?
0498 3004          JR      NC.049EH      :Ja: 'F1' - 'F4' ausblenden
049A D636          SUB     36H           :':' bis 'Z' wird zu 05H bis 24H
049C 180A          JR      04A8H         :Addiere C0H hinzu für Grafik

049E FE60          CP      60H           :'F1' bis 'F4' ausblenden
04A0 3808          JR      C.04AAH       :Sprung wenn 'F1' bis 'F4'
04A2 FE7B          CP      7BH           :Shift-'F1' bis Shift-'F4' ?
04A4 3004          JR      NC.04AAH      :Ja: keine Grafik
04A6 D63B          SUB     3BH           :''' bis 'z' wird zu 25H bis 3FH

04A8 C6C0          ADD     A.0C0H        :Grafikzeichen von C0H bis FFH

: In A ist jetzt der richtige Wert von 01H bis FFH

04AA 322440        LD      (4024H).A     ;Wert abspeichern für Repeat
04AD 57            LD      D.A           :Ablegen für Warteschleife
04AE 010020        LD      BC.2000H      :Warteschleife aufrufen zur
04B1 CD6000        CALL    0060H         :Verhinderung des Tastenprellens
04B4 7A            LD      A.D           :Wert zurück in A
04B5 FE0D          CP      0DH           :Wurde 'RETURN' gedrückt ?
04B7 2807          JR      Z.04C0H       :Ja: MOD SEL abschalten
04B9 FE01          CP      01H           :Wurde 'BREAK' gedrückt ?
04BB 2803          JR      Z.04C0H       :Ja: MOD SEL abschalten
04BD C0            RET     NZ            ;Fertig wenn nicht Break

04BE EF            RST     28H           :RST 28H wenn Break gedrückt
                                         ;wurde (für DOS)
04BF C9            RET

: MOD SEL aus
: (wenn 'RETURN' oder 'BREAK' gedrückt wurde)

04C0 211840        LD      HL.4018H      :MOD SEL abschalten
04C3 CBB6          RES     06H.(HL)
04C5 FE01          CP      01H           :'BREAK' gedrückt ?
04C7 18F4          JR      04BDH         :weiter bei 04BDH

: MOD SEL gedrückt

04C9 3E40          LD      A.40H         :MOD SEL umschalten
04CB AE            XOR     (HL)          :(Bit 6 der Adresse 4018H um-
04CC 77            LD      (HL).A        :schalten)
04CD AF            XOR     A             :Keinen Wert übergeben
04CE 18DD          JR      04ADH         :Routine abschliessen

: CTRL gedrückt

04D0 CBFE          SET     07H.(HL)      :CTRL Bit setzen
04D2 18F9          JR      04CDH         :und keinen Wert übergeben
```

```
: RPT gedrückt

  04D4 3A2440          LD      A,(4024H)    :Letzten Wert nach A laden
  04D7 18A6            JR      047FH        :und damit weiterrechnen

: POS errechnen (Fortsetzung von 0350H)

  04D9 ED52            SBC     HL,DE        :HL = Position des Cursors im
                                            :Bildschirm
  04DB 112800          LD      DE,0028H     :DE = 40 (Anzahl der Zeichen pro
                                            :Zeile)
  04DE B7              OR      A            :CY = 0
  04DF ED52            SBC     HL,DE        :Solange 40 von HL abziehen,
  04E1 30FB            JR      NC,04DEH     :bis HL < 0 ist
  04E3 19              ADD     HL,DE        :Dann 40 aufaddieren:
                                            :HL = Position des Cursors
                                            :in der Zeile
  04E4 7D              LD      A,L          :A = POS
  04E5 E1              POP     HL           :PTZ zurück
  04E6 C9              RET

: Druckerroutine (wird über DCB-Aufruf angesprochen)

  04E7 79              LD      A,C          :A = ASCII-Code des zu druckenden
                                            :Zeichens
  04E8 B7              OR      A            :Soll 00H ausgegeben werden ?
  04E9 283E            JR      Z,0529H      :Ja: Nur auf Drucker warten
  04EB FE0B            CP      0BH          :Seitenvorschub (VT) ?
  04ED 280A            JR      Z,04F9H      :Ja: Zeilenzähler mitzählen
  04EF FE0C            CP      0CH          :Seitenvorschub (FF) ?
  04F1 201B            JR      NZ,050EH     :Nein: Zeichen ausdrucken
  04F3 AF              XOR     A            :A = Maximale Anzahl von Zeilen
  04F4 DDB603          OR      (IX+03H)     :pro Seite. Ist A = 0 ?
  04F7 2815            JR      Z,050EH      :Ja: 00H zum Drucker geben
  04F9 DD7E03          LD      A,(IX+03H)   :A = max. Zeilenzahl pro Seite
  04FC DD9604          SUB     (IX+04H)     :minus der Anzahl bereits
                                            :gedruckter Zeilen auf der Seite,
                                            :ergibt die Anzahl der Zeilen bis
                                            :zur nächsten Seite
  04FF 47              LD      B,A          :Diese Zahl nach B
  0500 CD2905          CALL    0529H        :Drucker bereit ?
  0503 20FB            JR      NZ,0500H     :Nein: Auf Drucker warten
  0505 0E0A            LD      C,0AH        :Zeilenvorschübe (LF) ausgeben
  0507 CD3C05          CALL    053CH
  050A 10F4            DJNZ    0500H        :Bis zur nächsten Seite
  050C 1816            JR      0524H        :Zeilenzähler auf 0 setzen
```

```
; Zeichen ausdrucken

  050E CD2905          CALL   0529H          :Drucker bereit ?
  0511 20FB            JR     NZ,050EH       :Nein: Auf Drucker warten
  0513 CD3C05          CALL   053CH          :Zeichen in C ausgeben
  0516 FE0D            CP     0DH            :War es ein Zeilenvorschub ?
  0518 C0              RET    NZ             :Nein: Fertig
  0519 DD3404          INC    (IX+04H)       :Ja: Zeilenzähler erhöhen
  051C DD7E04          LD     A,(IX+04H)     :A = Anzahl der gedruckten Zeilen
  051F DDBE03          CP     (IX+03H)       :Mit Maximalzahl vergleichen
                                             :(Ist die Seite voll ?)
  0522 79              LD     A,C            :gedrucktes Zeichen zurück in A
  0523 C0              RET    NZ             :Nein: Fertig
  0524 DD360400        LD     (IX+04H),00H   :Ja: Zeilenzähler auf 0 für
                                             :neue Seite
  0528 C9              RET

; Drucker bereit ? (AF)
; I: -
; O: Z = 1 wenn Drucker bereit

  0529 3E07            LD     A,07H          :Register 7 des PSG
  052B D3F8            OUT    (0F8H),A       :anwählen
  052D 3E7F            LD     A,7FH          :I/O Port A auf Ausgabe und
  052F D3F9            OUT    (0F9H),A       :I/O Port B auf Eingabe stellen
  0531 3E0F            LD     A,0FH          :I/O Port B anwählen
  0533 D3F8            OUT    (0F8H),A       :(Reg. 15 des PSG)
  0535 DBF9            IN     A,(0F9H)       :und Druckerstatus holen
  0537 E6EF            AND    0EFH           :Statusbits maskieren
  0539 FE2F            CP     2FH            :und Z-Flag setzen
  053B C9              RET

; C zum Drucker geben ohne Überprüfung

  053C 3E07            LD     A,07H          :Register 7 des PSG anwählen
  053E D3F8            OUT    (0F8H),A
  0540 3E7F            LD     A,7FH          :I/O Port A auf Ausgabe und
  0542 D3F9            OUT    (0F9H),A       :I/O Port B auf Eingabe stellen
  0544 3E0E            LD     A,0EH          :Register 14 (I/O Port A)
  0546 D3F8            OUT    (0F8H),A       :anwählen
  0548 79              LD     A,C            :und Byte ausgeben
  0549 D3F9            OUT    (0F9H),A
  054B 3E07            LD     A,07H          :Jetzt beide Ports auf Ausgabe
  054D D3F8            OUT    (0F8H),A
  054F 3EFF            LD     A,0FFH
  0551 D3F9            OUT    (0F9H),A
  0553 3E0F            LD     A,0FH          :Zum Port B
  0555 D3F8            OUT    (0F8H),A
  0557 AF              XOR    A              :00H ausgeben
  0558 D3F9            OUT    (0F9H),A
  055A 3E0F            LD     A,0FH          :Dann nach Port B
  055C D3F8            OUT    (0F8H),A
  055E 3E01            LD     A,01H          :01H ausgeben und damit dem
                                             :Druckerinterface mitteilen, daß
                                             :ein Byte ausgegeben wurde
  0560 D3F9            OUT    (0F9H),A
  0562 79              LD     A,C            :Byte zurück nach A
  0563 C9              RET
```

```
: Ausgaberoutine (Fortsetzung von 0335H)

  0564 FA1F02        JP    M.021FH     :Ausgabe auf Cassette ?
                                       :Ja: weiter bei 021FH
  0567 C29C03        JP    NZ.039CH    :Ausgabe auf Drucker ?
                                       :Ja: weiter bei 039CH
  056A C33A03        JP    033AH       :Sonst Ausgabe auf Bildschirm

: Start 2 (Fortsetzung von 06ACH)

  056D 31F841        LD    SP.41F8H    :Stackpointer festlegen
  0570 AF            XOR   A           :Port 255 löschen (NBGRD. CHAR 1
  0571 D3FF          OUT   (0FFH).A    :und LGR ausführen)
  0573 2100F4        LD    HL.0F400H   :Farbspeicher löschen
  0576 1101F4        LD    DE.0F401H
  0579 01FF03        LD    BC.03FFH
  057C 3600          LD    (HL).00H
  057E EDB0          LDIR
  0580 0EFF          LD    C.0FFH      :C = Portadresse
  0582 ED78          IN    A.(C)       :Einstellungen für NTSC-Farbnorm
  0584 E608          AND   08H         :(wirkungslos in der PAL-Norm)
  0586 47            LD    B.A
  0587 D3FC          OUT   (0FCH).A
  0589 ED78          IN    A.(C)
  058B E608          AND   08H
  058D 211137        LD    HL.3711H    :--
  0590 110038        LD    DE.3800H    :DE = Zeiger auf
                                       :CRTC-Programmierungstabelle
                                       :für PAL-Norm
  0593 A8            XOR   B
  0594 00            NOP*
  0595 00            NOP
  0596 D3FD          OUT   (0FDH).A
  0598 ED78          IN    A.(C)
  059A E608          AND   08H
  059C 216935        LD    HL.3569H    :HL = Zeiger auf Farbcodetabelle
  059F A8            XOR   B
  05A0 1812          JR    05B4H       :weiter bei 05B4H

: Unbenutzte Routine (Nur für Geräte mit NTSC-Farbnorm)

  05A2 D3FE          OUT   (0FEH).A
  05A4 ED78          IN    A.(C)
  05A6 E608          AND   08H
  05A8 210A37        LD    HL.370AH
  05AB A8            XOR   B
  05AC 2006          JR    NZ.05B4H
  05AE 213137        LD    HL.3731H
  05B1 112338        LD    DE.3823H
```

```
; Start 3 (Fortsetzung von 05A0H)

  05B4 E5                  PUSH    HL            :HL retten
  05B5 EB                  EX      DE,HL         :HL = CRTC-Tabelle
  05B6 11F042              LD      DE,42F0H      :Tabelle ins RAM von 42F0H
  05B9 012300              LD      BC,0023H      :bis 4312H kopieren
                                                 :(Programmierungstabellen für den
                                                 :LGR- und FGR-Modus sowie die
                                                 :drei Zeitwerte für die
                                                 :Cassettenroutinen)
  05BC EDB0                LDIR
  05BE E1                  POP     HL            :HL zurück
  05BF 119043              LD      DE,4390H      :Auch die Farbcodetabelle ins
  05C2 011000              LD      BC,0010H      :RAM kopieren (16 Werte)
  05C5 EDB0                LDIR
  05C7 C36C00              JP      006CH         :weiter bei 006CH

; Reset 2 (Fortsetzung von 0069H)

  05CA AF                  XOR     A             :A = 00H
  05CB D3ED                OUT     (0EDH),A      :? (Port unbenutzt, DOS ?)
  05CD 3A04F8              LD      A,(0F804H)    :'R' gedrückt ?
  05D0 CB57                BIT     02H,A
  05D2 C20000              JP      NZ,0000H      :Ja: Kaltstart
  05D5 C3C006              JP      06C0H         :Nein: Warmstart


  05D8 FF                  RST     38H           :--

; Eingabe einer Zeile incl. Cursorsteuerung, FKey-Auswertung und
; Darstellung der eingegebenen Zeichen auf dem Bildschirm (AF,BC,DE,HL)
; Die Eingabe wird durch 'RETURN' oder 'BREAK' abgeschlossen
; Im Buffer ist das Ende durch 0DH gekennzeichnet
; I: HL = Adresse des Buffers in dem die eingegebenen Zeichen gespeichert
;         werden
;    B  = Maximale Anzahl der einzugebenden Zeichen
;         (mehr Zeichen werden nicht angenommen, die Eingabe muß durch 'ENTER'
;         oder 'BREAK' beendet werden
; O: HL = Anfangsadresse des Buffers (Zeiger auf das erste Zeichen)
;    DE = 401DH (vom 0033H Aufruf)
;    B  = Anzahl der tatsächlich eingegebenen Zeichen
;    C  = B vom Anfang
;    A  = Letztes Zeichen (0DH oder 01H)
;    CY = 1 falls die Eingabe durch 'BREAK' abgebrochen wurde

  05D9 E5                  PUSH    HL            :Bufferadresse retten
  05DA 3E0E                LD      A,0EH         :Cursor einschalten
  05DC CD3300              CALL    0033H
  05DF 48                  LD      C,B           :C = Max. Anzahl von Zeichen
  05E0 C30030              JP      3000H         :weiter bei 3000H zur FKey-
                                                 :Auswertung
```

```
05E3 FE20       CP      20H         :Kontrolltasten betätigt ?
05E5 3025       JR      NC,060CH    :Nein: Zeichen abspeichern
05E7 FE0D       CP      0DH         :'RETURN' ?
05E9 CA6206     JP      Z,0662H     :Ja: weiter bei 0662H
05EC FE1F       CP      1FH         :'CLEAR' ?
05EE 2829       JR      Z,0619H     :Ja: weiter bei 0619H
05F0 FE01       CP      01H         :'BREAK' ?
05F2 286D       JR      Z,0661H     :Ja: weiter bei 0661H
05F4 11E005     LD      DE,05E0H    :05E0H als Rücksprungadresse
05F7 D5         PUSH    DE          :im Stack ablegen
05F8 FE08       CP      08H         :Zeichen löschen (Linkspfeil) ?
05FA 2834       JR      Z,0630H     :Ja: weiter bei 0630H
05FC FE18       CP      18H         :Zeile löschen ?
                                    :(Shift-Linkspfeil)
05FE 282B       JR      Z,062BH     :Ja: weiter bei 062BH
0600 FE09       CP      09H -       :Vor bis zum nächsten TAB ?
                                    :(Rechtspfeil)
0602 2842       JR      Z,0646H     :Ja: weiter bei 0646H
0604 FE19       CP      19H         :Shift-Rechtspfeil ? (32 Zeichen
                                    :pro Zeile im GENIE I)
0606 2839       JR      Z,0641H     :Ja: weiter bei 0641H
0608 FE0A       CP      0AH         :Neue Zeile beginnen ?
                                    :(Tiefpfeil)
060A C0         RET     NZ          :Nein: Kontrollzeichen nicht
                                    :verwertbar: neues Zeichen holen
060B D1         POP     DE          :Rücksprungadresse aus Stack
                                    :nehmen
060C 77         LD      (HL),A      :Zeichen im Buffer ablegen
060D 78         LD      A,B         :Zeicheneingabe noch erlaubt ?
060E B7         OR      A
060F 28CF       JR      Z,05E0H     :Nein: neues Zeichen holen
0611 7E         LD      A,(HL)      :Zeichen aus Buffer holen
0612 23         INC     HL          :Bufferzeiger +1
0613 CD3300     CALL    0033H       :Zeichen darstellen
0616 05         DEC     B           :Zeichenzähler -1
0617 18C7       JR      05E0H       :nächstes Zeichen holen

: 'CLEAR' gedrückt

0619 CDC901     CALL    01C9H       :CLS aufrufen
061C 41         LD      B,C         :Zeichenzähler auf Neuanfang
061D E1         POP     HL          :HL auf Bufferstart
061E E5         PUSH    HL          :und wieder retten
061F C3E005     JP      05E0H       :neues Zeichen holen

: Zeile löschen

0622 CD3006     CALL    0630H       :Zeichen löschen und vom Buffer-
                                    :zeiger 1 abziehen
0625 2B         DEC     HL          :Bufferzeiger -1
0626 7E         LD      A,(HL)      :letztes Zeichen holen
0627 23         INC     HL          :Bufferzeiger wieder erhöhen
0628 FE0A       CP      0AH         :Wurde bereits eine neue Zeile
                                    :begonnen ?
062A C8         RET     Z           :Ja: dann nur bis dahin löschen
```

```
: Shift-Linkspfeil gedrückt

 062B 78          LD    A,B        ;Wurden bereits Zeichen
                                   ;eingegeben ? (Ist der Zeichen-
 062C B9          CP    C          ;zähler nicht mehr auf dem
                                   ;Anfangswert ?)
 062D 20F3        JR    NZ,0622H   ;Ja: Zeile löschen
 062F C9          RET              ;Nein: Fertig

: Linkspfeil gedrückt

 0630 78          LD    A,B        ;Ist der Zeichenzähler noch auf
 0631 B9          CP    C          ;dem Anfangswert ?
 0632 C8          RET   Z          ;Ja: Fertig
 0633 2B          DEC   HL         ;Bufferzeiger -1
 0634 7E          LD    A,(HL)     ;Voriges Zeichen holen
 0635 FE0A        CP    0AH        ;Wurde eine neue Zeile begonnen ?
 0637 23          INC   HL         ;Ja: Bufferzeiger +1
 0638 C8          RET   Z          ;und Rücksprung
 0639 2B          DEC   HL         ;Nein: Bufferzeiger -1
 063A 3E08        LD    A,08H      ;Backspace auch darstellen
 063C CD3300      CALL  0033H      ;(Zeichen auf Bildschirm löschen)
 063F 04          INC   B          ;Zeichenzähler +1
 0640 C9          RET

: Shift-Rechtspfeil gedrückt

 0641 3E17        LD    A,17H      ;17H war Code für Umschaltung
 0643 C33300      JP    0033H      ;zum 32 Zeichen Modus
                                   ;(wirkungslos im Colour Genie)

: Rechtspfeil gedrückt

 0646 CD4803      CALL  0348H      ;POS nach A holen
 0649 E607        AND   07H        ;Letzte TAB-Position errechnen
                                   ;(POS durch 8 teilbar machen)
 064B 2F          CPL              ;Zahl negieren
 064C 3C          INC   A          ;(2nd complement bilden)
 064D C608        ADD   A,08H      ;+8 ergibt Anzahl der Zeichen bis
                                   ;zur nächsten TAB-Position
 064F 5F          LD    E,A        ;E als Zähler benutzen
 0650 78          LD    A,B        ;Dürfen noch Zeichen eingefügt
 0651 B7          OR    A          ;werden ?
 0652 C8          RET   Z          ;Nein: Rücksprung
 0653 3E20        LD    A,20H      ;Ja: Leerzeichen im Buffer
 0655 77          LD    (HL),A     ;ablegen
 0656 23          INC   HL         ;und Bufferzeiger erhöhen
 0657 D5          PUSH  DE         ;DE retten
 0658 CD3300      CALL  0033H      ;Leerzeichen darstellen
 065B D1          POP   DE         ;DE zurück
 065C 05          DEC   B          ;Zeichenzähler -1
 065D 1D          DEC   E          ;TAB-Zähler -1
 065E C8          RET   Z          ;Rücksprung wenn TAB erreicht
 065F 18EF        JR    0650H      ;Sonst weiter einfügen
```

```
; 'BREAK' gedrückt

  0661 37            SCF                      ;CY = 1 setzen und wie 'RETURN'
                                              ;behandeln

; 'RETURN' gedrückt

  0662 F5            PUSH      AF             ;Flags retten (CY = 0 bei
                                              ;'RETURN')
  0663 3E0D          LD        A,0DH          ;A = ASCII-Code für RETURN
  0665 77            LD        (HL),A         ;Im Buffer ablegen
  0666 CD3300        CALL      0033H          ;und darstellen
  0669 3E0F          LD        A,0FH          ;Cursor wieder ausschalten
  066B CD3300        CALL      0033H
  066E 79            LD        A,C            ;Anzahl der eingegebenen
  066F 90            SUB       B              ;Zeichen ermitteln
  0670 47            LD        B,A            ;und in B ablegen
  0671 F1            POP       AF             ;Flags zurück
  0672 E1            POP       HL             ;HL auf Bufferanfang
  0673 C9            RET

; Start 1

  0674 3E08          LD        A,08H          ;CHAR 2, NBGRD und LGR
  0676 D3FF          OUT       (0FFH),A       ;einstellen
  0678 321C43        LD        (431CH),A      ;Portwert im RAM ablegen
  067B 21D206        LD        HL,06D2H       ;RST-Vektoren und DCBs aufbauen
  067E 110040        LD        DE,4000H       ;(4000H bis 4035H)
  0681 013600        LD        BC,0036H
  0684 EDB0          LDIR
  0686 3D            DEC       A              ;Solange wiederholen bis
  0687 3D            DEC       A              ;A = 00H ist
  0688 20F1          JR        NZ,067BH
  068A 0627          LD        B,27H          ;Von 4036H bis 405CH das RAM
  068C 12            LD        (DE),A         ;auf 00H setzen
  068D 13            INC       DE
  068E 10FC          DJNZ      068CH
  0690 21AB34        LD        HL,34ABH       ;Tabelle der FKey-Texte ins RAM
  0693 115043        LD        DE,4350H       ;kopieren
  0696 013800        LD        BC,0038H
  0699 EDB0          LDIR
  069B 3E01          LD        A,01H          ;SCALE auf 1 setzen
  069D 321443        LD        (4314H),A
  06A0 213039        LD        HL,3930H       ;Zeiger auf die Tabelle der
  06A3 228C43        LD        (438CH),HL     ;Colour-Basic Befehle im RAM
                                              ;ablegen
  06A6 21DB39        LD        HL,39DBH       ;Zeiger auf die Sprungadressen-
                                              ;tabelle
  06A9 228E43        LD        (438EH),HL     ;auch im RAM ablegen
  06AC C36D05        JP        056DH          ;weiter bei 056DH
```

```
: Test ob ROM-Cassette vorhanden ist

  06AF 3A00C0          LD    A,(0C000H)   ;Erstes Byte aus ROM holen
  06B2 FE43            CP    43H          ;Ist es ein 'C' ?
  06B4 CA01C0          JP    Z,0C001H     ;Ja: ROM-Programm starten
  06B7 3A00C0          LD    A,(0C000H)   ;oder ist es
  06BA FE44            CP    44H          ;ein 'D' ?
  06BC CA01C0          JP    Z,0C001H     ;auch dann ROM-Programm starten
  06BF C9              RET                ;Sonst: Rücksprung

: BASIC Warmstart

  06C0 3A00C0          LD    A,(0C000H)   ;Ist ROM-Cassette vorhanden ?
  06C3 FE43            CP    43H          ;(Erstes Zeichen = 'C' ?)
  06C5 CA01C0          JP    Z,0C001H     ;Ja: ROM-Programm starten
  06C8 C3AE19          JP    19AEH        ;Nein: Zurück ins BASIC

  06CB FF              RST   38H          ;--
  06CC FF              RST   38H
  06CD FF              RST   38H
  06CE FF              RST   38H
  06CF FF              RST   38H
  06D0 FF              RST   38H
  06D1 FF              RST   38H
```

```
; Dieser Teil wird von Start 1 ins RAM ab 4000H kopiert
; (nähere Erklärungen siehe RAM-Listing)

  06D2 C3961C          JP      1C96H
  06D5 C3781D          JP      1D78H
  06D8 C3901C          JP      1C90H
  06DB C3D925          JP      25D9H
  06DE C9              RET
  06DF 00              NOP
  06E0 00              NOP
  06E1 C9              RET
  06E2 00              NOP
  06E3 00              NOP
  06E4 FB              EI
  06E5 C9              RET
  06E6 00              NOP

  06E7 01
  06E8 E303
  06EA 00
  06EB 07
  06EC 40
  06ED 20
  06EE 49

  06EF 07
  06F0 E430
  06F1 00
  06F3 44
  06F4 01
  06F5 01
  06F6 03

  06F7 06
  06F8 E704
  06FA 43
  06FB 00
  06FC 00
  06FD 50
  06FE 52

  06FF C30050          JP      5000H
  0702 C7              RST     00H
  0703 00              NOP
  0704 00              NOP
  0705 3E00            LD      A,00H
  0707 C9              RET
```

```
; X = X + 0.5 (SNG)

  0708 218013          LD     HL,1380H      :HL : Konst. 0.5

; X = X + (HL)

  070B CDC209          CALL   09C2H         :BCDE = (HL)
  070E 1806            JR     0716H         :X = X + BCDE

; X = (HL) - X (SNG)

  0710 CDC209          CALL   09C2H         :BCDE = (HL)

; X = BCDE - X (SNG)

  0713 CD8209          CALL   0982H         :X = -X

; X = BCDE + X (SNG)

  0716 78              LD     A,B           :A = Exp (BCDE)
  0717 B7              OR     A             :BCDE = 0 ?
  0718 C8              RET    Z             :Ja: X ist Ergebnis
  0719 3A2441          LD     A,(4124H)     :A = Exp (X)
  071C B7              OR     A             :X = 0 ?
  071D CAB409          JP     Z,09B4H       :Ja: X = BCDE (BCDE ist Ergebnis)
  0720 90              SUB    B             :A = Exp (X) - Exp (BCDE)
                                            :Ist der größere Wert in X ?
  0721 300C            JR     NC,072FH      :Ja: weiter bei 072FH
                                            :Nein: Summanden vertauschen:
  0723 2F              CPL                  :A = -A ( A positiv machen)
  0724 3C              INC    A             :A ist die Differenz der Expo-
                                            :nenten
  0725 EB              EX     DE,HL         :DE retten
  0726 CDA409          CALL   09A4H         :(SP) = X (2. Summanden im Stack
                                            :ablegen)
  0729 EB              EX     DE,HL         :DE zurück
  072A CDB409          CALL   09B4H         :X = BCDE (X = 1. Summand)
  072D C1              POP    BC            :BCDE = 2. Summand
  072E D1              POP    DE
                                            :In X ist jetzt der Summand mit
                                            :dem größeren Exponenten.
                                            :In A ist die Differenz der
                                            :beiden Exponenten
  072F FE19            CP     19H           :Ist die Differenz beider
                                            :Exponenten größer als 24 (Ist
                                            :die Differenz beider Summanden
                                            :größer als 2 hoch 24)
  0731 D0              RET    NC            :Ja: BCDE ist zu klein, die Summe
                                            :würde X nicht verändern
  0732 F5              PUSH   AF            :Exponentendifferenz retten
  0733 CDDF09          CALL   09DFH         :Mantissen berichtigen
  0736 67              LD     H,A           :H,7 = 0, wenn Signs ungleich
  0737 F1              POP    AF            :Exponentendifferenz zurück
```

```
0738 CDD707         CALL    07D7H       :Durch verschieben von CDE um
                                        :A-Bits nach rechts. Mantisse von
                                        :BCDE auf gleichen Exp wie X
                                        :bringen
073B B4             OR      H           :A.7 = 0, wenn Signs ungleich
073C 212141         LD      HL,4121H    :HL = Zeiger auf X
073F F25407         JP      P,0754H     :Sprung wenn Signs ungleich
0742 CDB707         CALL    07B7H       :Mantissen addieren
0745 D29607         JP      NC,0796H    :Überlauf der Mantissen ?
                                        :Nein: weiter bei 0796H
0748 23             INC     HL          :Ja: HL zeigt auf Exp (X)
0749 34             INC     (HL)        :Exp (X) +1 (Überlauf auf Exp
                                        :übertragen)
074A CAB207         JP      Z,07B2H     :Exp-Überlauf ? Ja: OV-Error
074D 2E01           LD      L,01H       :Nein: Mantisse durch 2 teilen
074F CDEB07         CALL    07EBH       :da Exp erhöht wurde
0752 1842           JR      0796H       :CDE aufrunden und in X ablegen

: Signs ungleich

0754 AF             XOR     A           :B = -B
0755 90             SUB     B
0756 47             LD      B,A
0757 7E             LD      A,(HL)      :Mantissen subtrahieren
0758 9B             SBC     A,E
0759 5F             LD      E,A
075A 23             INC     HL
075B 7E             LD      A,(HL)
075C 9A             SBC     A,D
075D 57             LD      D,A
075E 23             INC     HL
075F 7E             LD      A,(HL)      :MSBs subtrahieren
0760 99             SBC     A,C
0761 4F             LD      C,A         :Unterlauf ?

: SFLOAT: CDEB in 2-Exp-Format umwandeln und in X ablegen
:         Die Mantisse wird solange um ein Bit nach links geschoben, bis das
:         höchste Bit = 1 und der Exponent möglichst klein ist

0762 DCC307         CALL    C,07C3H     :Bei Über- oder Unterlauf
                                        :Mantisse (CDEB) invertieren
0765 68             LD      L,B         :HL = LSBs
0766 63             LD      H,E         :Mantisse ist jetzt CDHL
0767 AF             XOR     A           :A = 00H
0768 47             LD      B,A         :B = Zähler
0769 79             LD      A,C         :A = MSB
076A B7             OR      A           :MSB = 0 ?
076B 2018           JR      NZ,0785H    :Nein: Bitweise weiterschieben
076D 4A             LD      C,D         :C <- D <- H <- L <- 00H
076E 54             LD      D,H         :(Mantisse byteweise links
                                        :schieben)
076F 65             LD      H,L         :bis MSB <> 00H
0770 6F             LD      L,A
```

```
0771 78                 LD      A,B
0772 D608               SUB     08H          :Zähler -8 (da 8 Bits geschoben
                                             :wurden)
0774 FEE0               CP      0E0H         :Zähler = -32 ?
0776 20F0               JR      NZ,0768H     :Nein: Weiterschieben
                                             :Ja: Es wurde 4 mal geschoben,
                                             :die Mantisse ist 0
                                             :-> Ergebnis = 0

; X = 0 (SNG,DBL)

0778 AF                 XOR     A            :A = 00H
0779 322441             LD      (4124H),A    :Exp (X) = 00H -> X = 0
077C C9                 RET

; Mantisse CDHL bitweise nach links schieben bis höchstes Bit = 1 ist

077D 05                 DEC     B            :Zähler -1 (da 1 Bit geschoben
                                             :wird)
077E 29                 ADD     HL,HL        :HL = HL * 2 (= links schieben)
077F 7A                 LD      A,D
0780 17                 RLA                  :D = D * 2, Überlauf in CY
0781 57                 LD      D,A
0782 79                 LD      A,C          :C = C + C + CY
0783 8F                 ADC     A,A
0784 4F                 LD      C,A
0785 F27D07             JP      P,077DH      :C.7 = 1 ?
                                             :(höchstes Bit in CDHL =1 ?)
                                             :Nein: Weiterschieben
0788 78                 LD      A,B          :A = negative Anzahl der
                                             :Verschiebungen
0789 5C                 LD      E,H          :Mantisse = CDEB
078A 45                 LD      B,L
078B B7                 OR      A            :Nichts verschoben ?
078C 2808               JR      Z,0796H      :Ja: weiter bei 0796H
078E 212441             LD      HL,4124H     :Nein: Exponenten ändern
0791 86                 ADD     A,(HL)       :A = neuer Exponent
0792 77                 LD      (HL),A       :Exp speichern
0793 30E3               JR      NC,0778H     :X = 0 setzen, wenn alter Exp zu
                                             :klein war
0795 C8                 RET     Z            :Fertig, wenn Exp = 0
```

```
: CDEB aufrunden und in X ablegen
: Exp(X) und Sign werden beibehalten

  0796 78           LD     A,B        ;A = LSB

  0797 212441       LD     HL,4124H   ;HL : Exp (X)
  079A B7           OR     A          ;A,7 = 1 ? (Höchstes Bit des LSBs
                                      ;gesetzt ?)
  079B FCA807       CALL   M,07A8H    ;Ja: CDE aufrunden
  079E 46           LD     B,(HL)     ;B = Exp (X)
  079F 23           INC    HL
  07A0 7E           LD     A,(HL)     ;A = Signflag
  07A1 E680         AND    80H        ;Signbit ausblenden
  07A3 A9           XOR    C          ;und mit C verknüpfen
  07A4 4F           LD     C,A        ;BCDE ist negativ wenn das Sign-
                                      ;flag = 00H war
  07A5 C3B409       JP     09B4H      ;X = BCDE

: CDE aufrunden und Überlauf nach Exp(X) schreiben

  07A8 1C           INC    E          ;LSB +1, Überlauf ?
  07A9 C0           RET    NZ         ;Nein: Zurück
  07AA 14           INC    D          ;Ja: Nächtes Byte +1
  07AB C0           RET    NZ         ;Zurück wenn kein Überlauf
  07AC 0C           INC    C          ;MSB +1, Überlauf ?
  07AD C0           RET    NZ         ;Nein: Zurück
  07AE 0E80         LD     C,80H      ;Ja: Höchstes Bit der Mantisse
                                      ;auf 1 setzen
  07B0 34           INC    (HL)       ;und Exp(X) erhöhen, Überlauf ?
  07B1 C0           RET    NZ         ;Nein: Zurück, sonst OV Error

: OV - Error

  07B2 1E0A         LD     E,0AH      ;E = Fehlercode für OV-Error
  07B4 C3A219       JP     19A2H      ;Zur Fehlerroutine

: CDE = CDE + (HL) (SNG) Festkomma- (Mantissen-) addition

  07B7 7E           LD     A,(HL)     ;A = (HL)
  07B8 83           ADD    A,E        ;A = E + (HL)
  07B9 5F           LD     E,A        ;E = E + (HL)
  07BA 23           INC    HL         ;nächstes Byte
  07BB 7E           LD     A,(HL)
  07BC 8A           ADC    A,D        ;jetzt mit CY addieren
  07BD 57           LD     D,A
  07BE 23           INC    HL         ;letztes Byte
  07BF 7E           LD     A,(HL)
  07C0 89           ADC    A,C
  07C1 4F           LD     C,A
  07C2 C9           RET
```

```
; Mantisse CDEB und Signflag (4125H) invertieren

  07C3 212541          LD      HL,4125H      :HL : Signflag
  07C6 7E              LD      A,(HL)        :Flag invertieren
  07C7 2F              CPL
  07C8 77              LD      (HL),A
  07C9 AF              XOR     A             :A = 00H
  07CA 6F              LD      L,A           :L = 00H
  07CB 90              SUB     B
  07CC 47              LD      B,A           :B = 00H - B
  07CD 7D              LD      A,L           :A = 00H
  07CE 9B              SBC     A,E           :E = 00H - E - CY
  07CF 5F              LD      E,A
  07D0 7D              LD      A,L           :desgl. mit D
  07D1 9A              SBC     A,D
  07D2 57              LD      D,A
  07D3 7D              LD      A,L           :und mit C
  07D4 99              SBC     A,C
  07D5 4F              LD      C,A
  07D6 C9              RET

; CDE um A Bits nach rechts schieben, B wird LSB

  07D7 0600            LD      B,00H         :Überlauf auf 00H
  07D9 D608            SUB     08H           :Noch mehr als 8 Verschiebungen ?
  07DB 3807            JR      C,07E4H       :Nein: Bitweise schieben
  07DD 43              LD      B,E           :Ja: Bytes schieben
  07DE 5A              LD      E,D
  07DF 51              LD      D,C
  07E0 0E00            LD      C,00H         :00H -> C -> D -> E -> B
  07E2 18F5            JR      07D9H         :Noch weiter verschieben ?

; Bitweise schieben

  07E4 C609            ADD     A,09H         :Subtraktion rückgängig machen
                                             :und 1 addieren, da vor der
                                             :Verschiebung der Zähler
                                             :erniedrigt wird
  07E6 6F              LD      L,A           :L = Zähler
  07E7 AF              XOR     A             :A = 00H
  07E8 2D              DEC     L             :Zähler -1, Zähler = 0 ?
  07E9 C8              RET     Z             :Ja: Zurück
  07EA 79              LD      A,C           :Nein: CDE um ein Bit nach rechts
                                             :schieben
  07EB 1F              RRA
  07EC 4F              LD      C,A
  07ED 7A              LD      A,D           :D schieben
  07EE 1F              RRA
  07EF 57              LD      D,A
  07F0 7B              LD      A,E           :E schieben
  07F1 1F              RRA
  07F2 5F              LD      E,A
  07F3 78              LD      A,B           :B schieben
  07F4 1F              RRA
  07F5 47              LD      B,A           :Überlauf in B
  07F6 18EF            JR      07E7H         :weiter verschieben ?
```

```
  07F8 00                                              :Konstante 1 (SNG)
  07F9 00
  07FA 00
  07FB 81

; SNG-Koeffiziententabelle für LOG-Funktion

  07FC 03                                              :3 Koeffizienten

  07FD AA                                              :0.598979 ca. 2 / ( 5*LOG(2) )
  07FE 56
  07FF 19
  0800 80

  0801 F1                                              :0.961471 ca. 2 / ( 3*LOG(2) )
  0802 22
  0803 76
  0804 80

  0805 45                                              :2.88539   =  2 / ( 1*LOG(2) )
  0806 AA
  0807 38
  0808 82

; X = LOG ( X )
; berechnet den natürlichen Logarithmus der Zahl in X
; I: X = Zahl (<> 0 )
; O: X = LOG (Zahl)

  0809 CD5509          CALL    0955H                   :TEST2
  080C B7              OR      A                       :Arg = 0 ?
  080D EA4A1E          JP      PE,1E4AH                :Ja: FC-Error
  0810 212441          LD      HL,4124H                :HL : Exp (Arg)
  0813 7E              LD      A,(HL)                  :A = Exp (Arg)
  0814 013580          LD      BC,8035H                :BCDE = 0.707107 = SQR(2)/2
  0817 11F304          LD      DE,04F3H
                                                       ;Umrechnung Arg = x * 2 hoch n
  081A 90              SUB     B                       :Exp (Arg) - 128 = n (B ist 128)
  081B F5              PUSH    AF                      :n retten
  081C 70              LD      (HL),B                  ;X = x (Exp auf 80H setzen)
                                                       :(2-Exponent entfernen)
  081D D5              PUSH    DE                      :BCDE retten
  081E C5              PUSH    BC
  081F CD1607          CALL    0716H                   :X = X + BCDE = X + 1/2 * SQR(2)
  0822 C1              POP     BC                      :BCDE zurück
  0823 D1              POP     DE
  0824 04              INC     B                       :Exp (BCDE) +1:
                                                       ;BCDE = BCDE * 2 = SQR (2)
```

```
0825 CDA208      CALL  08A2H      ;X = BCDE / X = SQR(2) / X
0828 21F807      LD    HL,07F8H   ;HL : 1.0
082B CD1007      CALL  0710H      ;X = (HL) - X = 1 - X
082E 21FC07      LD    HL,07FCH   ;HL = Zeiger auf Zahlentabelle
0831 CD9A14      CALL  149AH      ;Reihel berechnen
0834 018080      LD    BC,8080H   ;BCDE = -0.5
0837 110000      LD    DE,0000H
083A CD1607      CALL  0716H      ;X = BCDE + X = X - 1/2
083D F1          POP   AF         ;n zurück
083E CD890F      CALL  0F89H      ;X = X + n
                                  ;und mit LOG(2) multiplizieren

; X = X * LOG(2)

0841 013180      LD    BC,8031H   ;BCDE = 0.693147 = LOG(2)
0844 111872      LD    DE,7218H   ;

; SMUL: X = BCDE * X
; Multipliziert zwei Zahlen einfacher Genauigkeit
; I: BCDE = 1. Faktor
;    X    = 2. Faktor
; O: X    = Produkt

0847 CD5509      CALL  0955H      ;TEST2
084A C8          RET   Z          ;X = 0 ? -> Ergebnis = 0
084B 2E00        LD    L,00H      ;Flag = 00H (MUL-Kennung)
084D CD1409      CALL  0914H      ;Exponenten verarbeiten
0850 79          LD    A,C        ;CDE (1. Faktor) ab 414FH im RAM
                                  ;ablegen
0851 324F41      LD    (414FH),A
0854 EB          EX    DE,HL
0855 225041      LD    (4150H),HL
0858 010000      LD    BC,0000H   ;BCDE = 00000000H
085B 50          LD    D,B        ;(In BCDE wird das Produkt
085C 58          LD    E,B        ;gebildet)
085D 216507      LD    HL,0765H   ;Letztes RET nach 0765H
0860 E5          PUSH  HL
0861 216908      LD    HL,0869H   ;Zweimal RET auf 0869H setzen
0864 E5          PUSH  HL         ;(0869H wird sofort und zweimal
0865 E5          PUSH  HL         ;nach RET, also insgesamt dreimal
                                  ;durchlaufen)
                                  ;(die Mantisse hat 3 Bytes !)
0866 212141      LD    HL,4121H   ;HL = Zeiger auf 2. Faktor
0869 7E          LD    A,(HL)     ;A = nächstes Byte der Mantisse
                                  ;des 2. Faktors
086A 23          INC   HL         ;Zeiger +1
086B B7          OR    A          ;Byte = 00H ?
086C 2824        JR    Z,0892H    ;Ja: weiter bei 0892H (A = 00H)
086E E5          PUSH  HL         ;Zeiger retten
086F 2E08        LD    L,08H      ;L = Zähler für 8 Bits
```

```
  0871 1F                 RRA                :Nächstes Bit in CY schieben
  0872 67                 LD      H,A        :Byte in H retten
  0873 79                 LD      A,C        :A = MSB
  0874 300B               JR      NC,0881H   :Sprung wenn Bit nicht gesetzt
  0876 E5                 PUSH    HL         :HL retten
  0877 2A5041             LD      HL,(4150H) :CDE = CDE + 1. Faktor
  087A 19                 ADD     HL,DE
  087B EB                 EX      DE,HL
  087C E1                 POP     HL         :HL zurück
  087D 3A4F41             LD      A,(414FH)  :
  0880 89                 ADC     A,C        :A = MSB
  0881 1F                 RRA                :CDEB um 1 Bit nach rechts
  0882 4F                 LD      C,A        :schieben
  0883 7A                 LD      A,D        :D schieben
  0884 1F                 RRA
  0885 57                 LD      D,A -
  0886 7B                 LD      A,E        :E schieben
  0887 1F                 RRA
  0888 5F                 LD      E,A
  0889 78                 LD      A,B        :B schieben
  088A 1F                 RRA
  088B 47                 LD      B,A
  088C 2D                 DEC     L          :Zähler -1
  088D 7C                 LD      A,H        :Byte zurück nach A
  088E 20E1               JR      NZ,0871H   :nächstes Bit prüfen
  0890 E1                 POP     HL         :Zeiger auf X zurück nach HL
  0891 C9                 RET                :Zweimal RET auf 0896H
                                             :zuletzt auf 0765H

; UPRO für SMUL
; CDEB um 1 Byte nach rechts schieben und mit 00H auffüllen

  0892 43                 LD      B,E        :00H -> C -> D -> E -> B
  0893 5A                 LD      E,D
  0894 51                 LD      D,C
  0895 4F                 LD      C,A
  0896 C9                 RET

; SDIV10: X = X / 10
; Dividiert Zahl in X durch 10
; I: X = Zahl einfacher Genauigkeit
; O: X = Zahl / 10

  0897 CDA409             CALL    09A4H      :(SP) = X
  089A 21D80D             LD      HL,0DD8H   :HL : 10

; X = (SP) / (HL)

  089D CDB109             CALL    09B1H      :X = BCDE = (HL)

; X = (SP) / X

  08A0 C1                 POP     BC         :BCDE = (SP)
  08A1 D1                 POP     DE
```

```
; SDIV: X = BCDE / X
; Dividiert zwei Zahlen einfacher Genauigkeit
; I: BCDE = Dividend
;    X    = Divisor
; O: X    = Quotient

 08A2 CD5509        CALL   0955H       :TEST2
 08A5 CA9A19        JP     Z,199AH     :/0-Error, wenn X = 0
 08A8 2EFF          LD     L,0FFH      :Flag = FFH (DIV-Kennung)
 08AA CD1409        CALL   0914H       :Exponenten und Vorzeichen ver-
                                       :rechnen
 08AD 34            INC    (HL)        :Exponenten berichtigen
 08AE 34            INC    (HL)
 08AF 2B            DEC    HL          :HL zeigt auf MSB (X)
 08B0 7E            LD     A,(HL)      :Divisor im RAM ablegen:
 08B1 328940        LD     (4089H),A   :MSB nach 4089H
 08B4 2B            DEC    HL
 08B5 7E            LD     A,(HL)
 08B6 328540        LD     (4085H),A   :1. LSB nach 4085H
 08B9 2B            DEC    HL
 08BA 7E            LD     A,(HL)
 08BB 328140        LD     (4081H),A   :2. LSB nach 4081H
 08BE 41            LD     B,C         :BHL = CDE
 08BF EB            EX     DE,HL       :BHL = Mantisse des Dividenden
 08C0 AF            XOR    A           :A = 00H
 08C1 4F            LD     C,A         :CDE = 000000H
 08C2 57            LD     D,A         :(In CDE wird das Ergebnis be-
 08C3 5F            LD     E,A         :rechnet)
 08C4 328C40        LD     (408CH),A   :(408CH) = 00H
                                       :MSB des Dividenden auf 0 setzen

 08C7 E5            PUSH   HL          :Dividend retten
 08C8 C5            PUSH   BC
 08C9 7D            LD     A,L         :A = 2.LSB des Dividenden
 08CA CD8040        CALL   4080H       :BHL = BHL - X (Mantissen von
                                       :Dividend und Divisor sub-
                                       :trahieren)
                                       :A = (408CH)
 08CD DE00          SBC    A,00H       :A = A - CY (Unterlauf der
                                       :letzten Subtraktion auch vom MSB
                                       :abziehen)
                                       :Unterlauf ?
 08CF 3F            CCF                :CY invertieren
 08D0 3007          JR     NC,08D9H    :Ja: weiter bei 08D9H
 08D2 328C40        LD     (408CH),A   :Nein: MSB zurückschreiben
 08D5 F1            POP    AF          :Dividend vom Stack löschen
 08D6 F1            POP    AF
 08D7 37            SCF                :CY = 1 -> Sprung wird nicht
 08D8 D2C1E1        JP     NC,0E1C1H   :ausgeführt
*08D9   C1          POP    BC          :Dividend zurück nach BHL
*08DA     E1        POP    HL          :(CY = 0 wegen 08D0H !)
 08DB 79            LD     A,C         :A = MSB des Ergebnisses
```

```
 08DC 3C               INC    A              :A.7 = 1 ?
 08DD 3D               DEC    A              :(S-Flag wird beeinflusst)
 08DE 1F               RRA                   :CY für die Rundungsroutine nach
                                             :A.7 schieben
 08DF FA9707           JP     M.0797H        :Ja: Fertig. CDE aufrunden (falls
                                             :CY = 1 war) und als Ergebnis
                                             :nach X schreiben
 08E2 17               RLA                   :A.7 nach CY zurückschieben
                                             :und CY ins Ergebnis einschieben
 08E3 7B               LD     A.E            :(C <- D <- E <- CY)
 08E4 17               RLA
 08E5 5F               LD     E.A
 08E6 7A               LD     A.D
 08E7 17               RLA
 08E8 57               LD     D.A
 08E9 79               LD     A.C
 08EA 17               RLA
 08EB 4F               LD     C.A
 08EC 29               ADD    HL.HL          :Dividend auch ein Bit nach links
 08ED 78               LD     A.B            :verschieben
 08EE 17               RLA                   :Überlauf von HL in B
 08EF 47               LD     B.A            :übernehmen
 08F0 3A8C40           LD     A.(408CH)      :Überlauf der Dividendenver-
 08F3 17               RLA                   :schiebung nach 408CH über-
 08F4 328C40           LD     (408CH).A      :nehmen (MSB des Dividenden)
 08F7 79               LD     A.C            :CDE = 000000H ?
 08F8 B2               OR     D              :(Ist das Ergebnis noch 0 ?)
 08F9 B3               OR     E
 08FA 20CB             JR     NZ.08C7H       :Nein: nächstes Bit verrechnen
 08FC E5               PUSH   HL             :Ja: HL retten
 08FD 212441           LD     HL.4124H       :HL : Exp(X)
 0900 35               DEC    (HL)           :X = X / 2
                                             :Unterlauf ?
 0901 E1               POP    HL             :HL zurück
 0902 20C3             JR     NZ.08C7H       :Nein: nächstes Bit verrechnen
 0904 C3B207           JP     07B2H          :Ja: OV-Error

; UPRO für SMUL. SDIV. DMUL und DDIV
; Exponenten und Vorzeichen verrechnen

; Einsprung für DDIV

 0907 3EFF             LD     A.0FFH         :Flag = FFH
 0909 2EAF             LD     L.0AFH         :--

; Einsprung für DMUL

*090A   AF             XOR    A              :Flag = 00H
 090B 212D41           LD     HL.412DH       :HL : MSB (Y)
 090E 4E               LD     C.(HL)         :C = MSB (Y)
 090F 23               INC    HL             :HL : Exp (Y)
 0910 AE               XOR    (HL)           :DMUL : A = Exp (Y)
                                             :DDIV : A = -Exp (Y) - 1
 0911 47               LD     B.A            :B = Exp
 0912 2E00             LD     L.00H          :weiter wie bei SMUL
```

```
: Einsprung für SMUL (L=00H) und SDIV (L=FFH)

  0914 78              LD     A,B        ;A = Exp (BCDE)
  0915 B7              OR     A          ;BCDE = 0 ?
  0916 281F            JR     Z,0937H    ;Ja: Ergebnis = 0
  0918 7D              LD     A,L        ;A = Flag
  0919 212441          LD     HL,4124H   ;HL : Exp (X)
  091C AE              XOR    (HL)       ;SMUL: A = Exp (X)
                                         ;SDIV: A = -Exp (X) - 1
  091D 80              ADD    A,B        ;SMUL: A = Exp (Y) + Exp (X)
                                         ;SDIV: A = Exp (Y) - Exp (X) - 1
  091E 47              LD     B,A        ;B = neuer Exp
  091F 1F              RRA               ;CY nach A.7 schieben
  0920 A8              XOR    B          ;mit B verknüpfen
                                         ;A.7 = 0 wenn es bei der Exp-
                                         ;Addition zu einem Über- bzw.
                                         ;Unterlauf nach CY und A.7 kam
                                         ;(Wenn beide Exponenten positiv
                                         ;waren (d.h. > 80H) kommt es nur
                                         ;zu einem Überlauf nach CY.
                                         ;A.7 wird 0 !)
  0921 78              LD     A,B        ;A = neuer Exp
  0922 F23609          JP     P,0936H    ;Weiter bei 0936H wenn es zu
                                         ;einem Über- bzw. Unterlauf kam
  0925 C680            ADD    A,80H      ;Offset zum Exp addieren
                                         ;(Exp steht jetzt richtig)
  0927 77              LD     (HL),A     ;neuen Exp in Exp (X) speichern
  0928 CA9008          JP     Z,0890H    ;Exp = 0 ? Ja: Fertig
  092B CDDF09          CALL   09DFH      ;Mantissen berichtigen,
                                         ;Vorzeichen ausblenden
  092E 77              LD     (HL),A     ;Signflag retten
  092F 2B              DEC    HL         ;HL = 4124H
  0930 C9              RET

: Einsprung von EXP ( X ) wenn X > 127 oder LSB(X) > 7DH
: wenn X < 0, dann X = 0 setzen sonst OV-Error

  0931 CD5509          CALL   0955H      :TEST2
  0934 2F              CPL               ;A positiv wenn X < 0
                                         ;sonst A negativ
  0935 E1              POP    HL         ;RET-Adresse löschen
  0936 B7              OR     A          ;Flags setzen
  0937 E1              POP    HL         ;RET-Adresse löschen
  0938 F27807          JP     P,0778H    ;A positiv: X = 0
  093B C3B207          JP     07B2H      ;sonst OV-Error
```

```
; SMUL10: X = X * 10
; Multipliziert Zahl in X mit 10
; I: X = Zahl einfacher Genauigkeit
; O: X = Zahl * 10

  093E CDBF09          CALL   09BFH        ;BCDE = X
  0941 78              LD     A,B          ;BCDE = 0 ?
  0942 B7              OR     A
  0943 C8              RET    Z            ;Ja: Ergebnis = 0
  0944 C602            ADD    A,02H        ;Exp +2 -> BCDE = BCDE * 4
  0946 DAB207          JP     C,07B2H      ;OV-Error bei Exp Überlauf
  0949 47              LD     B,A          ;Exp in BCDE setzen
  094A CD1607          CALL   0716H        ;X = X + BCDE = X + 4*X = 5 * X
  094D 212441          LD     HL,4124H     ;HL : Exp(X)
  0950 34              INC    (HL)         ;Exp (X) +1 -> X = X * 2
  0951 C0              RET    NZ           ;Ok wenn kein Überlauf
  0952 C3B207          JP     07B2H        ;OV-Error bei Exp Überlauf

; TEST2: SGN-Funktion für X
; Testet Zahl einfacher oder doppelter Genauigkeit wie der SGN-Befehl
; I: X = zu testende Zahl einfacher oder doppelter Genauigkeit
; O: wenn X < 0: A = FFH, CY = 1, Z = 0, S = 1
;    wenn X = 0: A = 00H, CY = 0, Z = 1, S = 0, P/V = 1
;    wenn X > 0: A = 01H, CY = 0, Z = 0, S = 0

  0955 3A2441          LD     A,(4124H)    ;A = Exp ( X )
  0958 B7              OR     A            ;X = 0 ?
  0959 C8              RET    Z            ;Ja: Fertig
  095A 3A2341          LD     A,(4123H)    ;A,7 = Sign
  095D FE2F            CP     2FH          ;--
 *095E   2F            CPL                 ;Einsprung von Vergleichsroutinen
  095F 17              RLA                 ;CY = Sign
  0960 9F              SBC    A,A          ;A = FFH wenn X < 0 sonst A = 00H
  0961 C0              RET    NZ           ;A ok wenn X < 0
  0962 3C              INC    A            ;X > 0: A = 01H
  0963 C9              RET

; FLOATA: A in eine Zahl einfacher Genauigkeit umwandeln
; I: A = Zahl
; O: X = Zahlenwert von A in einfacher Genauigkeit

  0964 0688            LD     B,88H        ;B = Exp für 2 hoch 8
  0966 110000          LD     DE,0000H     ;LSBs = 0
```

```
; FLOAT: Binärwert in eine Zahl einfacher Genauigkeit umwandeln
; 1. Binärzahl mit 8 Bit:
;    I: B  = 88H (Exp für 2 hoch 8)
;       A  = Binärwert
;       DE = 0000H
; 2. Binärzahl mit 16 Bit: (siehe auch CINT)
;    I: B  = 90H (Exp für 2 hoch 16)
;       A  = MSB des 16 Bit-Werts
;       D  = LSB des 16 Bit-Werts
;       E  = 00H
; 3. Binärzahl mit 24 Bit:
;    I: B  = 98H (Exp für 2 hoch 24)
;       A  = MSB des 24 Bit-Werts
;       DE = LSBs des 24 Bit-Werts
;
; O: X = Zahl in einfacher Genauigkeit

  0969 212441          LD     HL,4124H     ;HL : Exp (X)
  096C 4F              LD     C,A          ;C = 8-Bit Wert
  096D 70              LD     (HL),B       ;Exp setzen
  096E 0600            LD     B,00H        ;B = 00H (wird LSB bei 0762H)
  0970 23              INC    HL           ;HL : Signflag
  0971 3680            LD     (HL),80H     ;Signflag auf 80H setzen
  0973 17              RLA                 ;Signbit nach CY (für 0762H)
  0974 C36207          JP     0762H        ;und nach SFLOAT springen

; X = ABS ( X )

  0977 CD9409          CALL   0994H        ;TEST1
  097A F0              RET    P            ;Fertig wenn X > 0

; X = -X Typrichtig

  097B E7              RST    20H          ;TSTTYP
  097C FA5B0C          JP     M,0C5BH      ;nach 0C5BH wenn X INT
  097F CAF60A          JP     Z,0AF6H      ;TM-Error wenn X STR
                                           ;sonst X = -X

; SDNEG: X = -X
; Zahl in X negieren
; I: X = Zahl einfacher oder doppelter Genauigkeit
; O: X = negierte Zahl

  0982 212341          LD     HL,4123H     ;(HL) = MSB
  0985 7E              LD     A,(HL)       ;Sign umkehren
  0986 EE80            XOR    80H
  0988 77              LD     (HL),A
  0989 C9              RET
```

```
; X = SGN ( X )

  098A CD9409          CALL    0994H          ;TEST1
  098D 6F              LD      L,A            ;L = SGN-Wert (FFH, 00H oder 01H)
  098E 17              RLA                    ;Höchstes Bit nach CY schieben
  098F 9F              SBC     A,A            ;A = 00H bei positivem Ergebnis
                                              ;sonst A = FFH
  0990 67              LD      H,A            ;damit ist HL der INT-Wert von A
  0991 C39A0A          JP      0A9AH          ;HL in X ablegen

; TEST1: wie SGN-Funktion, aber Resultat in A
; Testet Zahl in X ob kleiner, gleich oder größer Null
; TM-Error wenn STR in X
; I: X = zu testende Zahl
; O: wenn X < 0: A = FFH, CY = 1, Z = 0, S = 1
;    wenn X = 0: A = 00H, CY = 0, Z = 1, S = 0, P/V = 1
;    wenn X > 0: A = 01H, CY = 0, Z = 0, S = 0

  0994 E7              RST     20H            ;TSTTYP
  0995 CAF60A          JP      Z,0AF6H        ;TM-Error wenn STR in X
  0998 F25509          JP      P,0955H        ;TEST2 wenn X SNG oder DBL
  099B 2A2141          LD      HL,(4121H)     ;HL = INT-Wert
  099E 7C              LD      A,H            ;HL = 0 ?
  099F B5              OR      L
  09A0 C8              RET     Z              ;Ja: ok
  09A1 7C              LD      A,H            ;A.7 = 1 wenn HL<0 sonst A.7 = 0
  09A2 18BB            JR      095FH          ;In TEST2 springen

; (SP) = X (SNG)
; (Wert in X retten)

  09A4 EB              EX      DE,HL          ;HL retten
  09A5 2A2141          LD      HL,(4121H)     ;HL = LSB (X)
  09A8 E3              EX      (SP),HL        ;LSBs in Stack bringen
  09A9 E5              PUSH    HL             ;und Rücksprungadresse retten
  09AA 2A2341          LD      HL,(4123H)     ;desgl. mit MSB und Exp
  09AD E3              EX      (SP),HL        ;
  09AE E5              PUSH    HL             ;
  09AF EB              EX      DE,HL          ;HL zurück
  09B0 C9              RET

; X = BCDE = (HL) (SNG)

  09B1 CDC209          CALL    09C2H          ;BCDE = (HL)

; X = BCDE (SNG)

  09B4 EB              EX      DE,HL          ;HL = LSBs
  09B5 222141          LD      (4121H),HL     ;nach X bringen
  09B8 60              LD      H,B            ;HL = MSB und Exp
  09B9 69              LD      L,C
  09BA 222341          LD      (4123H),HL     ;nach X bringen
  09BD EB              EX      DE,HL          ;HL zurück
  09BE C9              RET
```

```
: BCDE = X (SNG)

  09BF 212141            LD      HL,4121H     ;HL = Zeiger auf X

: BCDE = (HL) (SNG)

  09C2 5E                LD      E,(HL)       ;BCDE mit (HL) laden
  09C3 23                INC     HL
  09C4 56                LD      D,(HL)
  09C5 23                INC     HL
  09C6 4E                LD      C,(HL)
  09C7 23                INC     HL
  09C8 46                LD      B,(HL)
  09C9 23                INC     HL
  09CA C9                RET

: (HL) = X (SNG)

  09CB 112141            LD      DE,4121H     ;DE = Zeiger auf X
  09CE 0604              LD      B,04H        ;4 Bytes SNG-Wert
  09D0 1805              JR      09D7H        ;in Kopierroutine springen

: (DE) = (HL) Kopiert Speicher von (HL) nach (DE), Zähler = (40AFH)

  09D2 EB                EX      DE,HL        :

: wie oben aber von (DE) nach (HL)

  09D3 3AAF40            LD      A,(40AFH)    ;A = VT = Zähler

: Kopierroutine kopiert A Bytes von (DE) nach (HL)

  09D6 47                LD      B,A          ;B = Zähler
  09D7 1A                LD      A,(DE)       ;Byte von (DE) holen
  09D8 77                LD      (HL),A       ;und in (HL) ablegen
  09D9 13                INC     DE           ;Zeiger +1
  09DA 23                INC     HL
  09DB 05                DEC     B            ;Zähler -1
  09DC 20F9              JR      NZ,09D7H     ;Fertig ?
  09DE C9                RET
```

```
: Vorzeichenbehandlung für Grundrechenarten
: Mantissen von X und BCDE richtig stellen (Höchstes Bit auf 1)
: und Vorzeichen verrechnen
: I: X     = Zahl einfacher oder doppelter Genauigkeit
:    BCDE = wie X
: O: Mantisse von X und BCDE richtig (höchstes Bit = 1)
:    A.7 = 1 wenn Signs beider Zahlen identisch, sonst A.7 = 0

  09DF 212341        LD     HL,4123H     ;HL = MSB von X
  09E2 7E            LD     A,(HL)       ;A = MSB von X
  09E3 07            RLCA                ;CY und A.0 = Sign von X
  09E4 37            SCF                 ;CY = 1
  09E5 1F            RRA                 ;A.7 = CY = 1, CY = A.0 = Sign
  09E6 77            LD     (HL),A       ;Mantisse richtig stellen
  09E7 3F            CCF                 ;CY invertieren
  09E8 1F            RRA                 ;A.0 = invertiertes Sign
  09EA 23            INC    HL
  09E9 23            INC    HL           ;HL = 4125H (Signflag)
  09EB 77            LD     (HL),A       ;Signflag retten
  09EC 79            LD     A,C          ;desgl. mit BCDE aber Sign lassen
  09ED 07            RLCA
  09EE 37            SCF
  09EF 1F            RRA
  09F0 4F            LD     C,A
  09F1 1F            RRA
  09F2 AE            XOR    (HL)         ;Signs verknüpfen
  09F3 C9            RET

: X = Y (SNG,DBL)

  09F4 212741        LD     HL,4127H     ;HL = Zeiger auf Y

: X = (HL) (SNG,DBL)

  09F7 11D209        LD     DE,09D2H     ;DE = Adresse der Kopierroutine
                                         ;(HL) -> (DE)
  09FA 1806          JR     0A02H        ;weiter bei 0A02H

: Y = X (SNG,DBL)

  09FC 212741        LD     HL,4127H     ;HL = Zeiger auf Y

: (HL) = X (SNG,DBL)

  09FF 11D309        LD     DE,09D3H     ;DE = Adresse der Kopierroutine
                                         ;(DE) -> (HL)
  0A02 D5            PUSH   DE           ;Adresse für RET in Stack
  0A03 112141        LD     DE,4121H     ;DE = Zeiger auf X (SNG)
  0A06 E7            RST    20H          ;TSTTYP
  0A07 D8            RET    C            ;DE ok wenn X SNG
  0A08 111D41        LD     DE,411DH     ;sonst DE = Zeiger auf X (DBL)
  0A0B C9            RET
```

```
; CP X , BCDE (SNG)
; I: -
; O: wenn X < BCDE: A = FFH, CY = 1, Z = 0, S = 1
;    wenn X = BCDE: A = 00H, CY = 0, Z = 1, S = 0
;    wenn X > BCDE: A = 01H, CY = 0, Z = 0, S = 0

  0A0C 78            LD     A,B          ;BCDE = 0 ?
  0A0D B7            OR     A
  0A0E CA5509        JP     Z,0955H      ;Ja: TEST2
  0A11 215E09        LD     HL,095EH     ;RET-Adresse auf TEST2 legen
  0A14 E5            PUSH   HL
  0A15 CD5509        CALL   0955H        ;TEST2
  0A18 79            LD     A,C          ;A = MSB (BCDE)
  0A19 C8            RET    Z            ;Nach TEST2, wenn X = 0
                                         ;(nur BCDE testen)
  0A1A 212341        LD     HL,4123H     ;HL : MSB (X)
  0A1D AE            XOR    (HL)         ;Signs vergleichen
  0A1E 79            LD     A,C          ;A = MSB (BCDE)
  0A1F F8            RET    M            ;nach TEST2, wenn Signs ungleich
                                         ;(nur BCDE testen)
  0A20 CD260A        CALL   0A26H        ;X und BCDE Byte für Byte
                                         ;vergleichen
  0A23 1F            RRA                 ;A.7 = CY
  0A24 A9            XOR    C            ;mit Vorzeichen von BCDE ver-
                                         ;knüpfen
  0A25 C9            RET                 ;und weiter bei TEST2

; UPRO für CP (Vergleicht BCDE mit X Byte für Byte)

  0A26 23            INC    HL           ;Exp vergleichen
  0A27 78            LD     A,B
  0A28 BE            CP     (HL)
  0A29 C0            RET    NZ           ;RET wenn ungleich
  0A2A 2B            DEC    HL           ;Desgl. mit MSBs
  0A2B 79            LD     A,C
  0A2C BE            CP     (HL)
  0A2D C0            RET    NZ
  0A2E 2B            DEC    HL           ;1. LSB
  0A2F 7A            LD     A,D
  0A30 BE            CP     (HL)
  0A31 C0            RET    NZ
  0A32 2B            DEC    HL           ;2. LSB
  0A33 7B            LD     A,E
  0A34 96            SUB    (HL)         ;SUBtraktion, damit A = 00H wenn
                                         ;E gleich (HL)
  0A35 C0            RET    NZ
  0A36 E1            POP    HL           ;RET-Adresse entfernen (0A23H)
  0A37 E1            POP    HL           ;RET-Adresse entfernen (095EH)
  0A38 C9            RET                 ;Zurück mit A = 00H, Z = 1
```

```
; CP HL , DE (INT)
; I: -
; O: wenn HL < DE: A = FFH, CY = 1, Z = 0, S = 1
;    wenn HL = DE: A = 00H, CY = 0, Z = 1, S = 0
;    wenn HL > DE: A = 01H, CY = 0, Z = 0, S = 0

  0A39 7A          LD     A,D        ;Signs vergleichen
  0A3A AC          XOR    H
  0A3B 7C          LD     A,H        ;A = MSB (HL)
  0A3C FA5F09      JP     M,095FH    ;Nach TEST2, wenn Signs ungleich
  0A3F BA          CP     D          ;MSBs vergleichen
  0A40 C26009      JP     NZ,0960H   ;Nach TEST2 wenn ungleich
  0A43 7D          LD     A,L        ;LSBs vergleichen
  0A44 93          SUB    E          ;SUBtraktion, damit A = 00H wenn
                                     ;L gleich E
  0A45 C26009      JP     NZ,0960H   ;Nach TEST2 wenn LSBs ungleich
  0A48 C9          RET

; CP X , (DE) (DBL)
; I:-
; O: wenn X < (DE): A = FFH, CY = 1, Z = 0, S = 1
;    wenn X = (DE): A = 00H, CY = 0, Z = 1, S = 0
;    wenn X > (DE): A = 01H, CY = 0, Z = 0, S = 0

  0A49 212741      LD     HL,4127H   ;HL : Zeiger auf Y
  0A4C CDD309      CALL   09D3H      ;Kopiere (DE) nach (HL)
  0A4F 112E41      LD     DE,412EH   ;DE : Exp (Y)
  0A52 1A          LD     A,(DE)     ;A = Exp (Y)
  0A53 B7          OR     A          ;Y = 0 ?
  0A54 CA5509      JP     Z,0955H    ;Ja: TEST2 ausführen
  0A57 215E09      LD     HL,095EH   ;HL = RET-Adresse nach TEST2
  0A5A E5          PUSH   HL
  0A5B CD5509      CALL   0955H      ;TEST2
  0A5E 1B          DEC    DE         ;DE : MSB (Y)
  0A5F 1A          LD     A,(DE)     ;A = MSB (Y)
  0A60 4F          LD     C,A        ;C = MSB (Y) (für TEST2)
  0A61 C8          RET    Z          ;TEST2 mit Y wenn X = 0
  0A62 212341      LD     HL,4123H   ;HL : MSB (X)
  0A65 AE          XOR    (HL)       ;Signs vergleichen
  0A66 79          LD     A,C        ;A = MSB (Y)
  0A67 F8          RET    M          ;TEST2 mit Y wenn Signs ungleich
  0A68 13          INC    DE         ;DE : Exp (Y)
  0A69 23          INC    HL         ;HL : Exp (X)
  0A6A 0608        LD     B,08H      ;B = Zähler für 8 Bytes
  0A6C 1A          LD     A,(DE)     ;X und Y Byte für Byte
  0A6D 96          SUB    (HL)       ;vergleichen
  0A6E C2230A      JP     NZ,0A23H   ;weiter bei 0A23H wenn ungleich
  0A71 1B          DEC    DE         ;sonst Zeiger +1
  0A72 2B          DEC    HL
  0A73 05          DEC    B          ;Zähler -1
  0A74 20F6        JR     NZ,0A6CH   ;Nächstes Byte vergleichen
  0A76 C1          POP    BC         ;RET-Adresse löschen (095EH)
  0A77 C9          RET
```

```
; CP X , Y (DBL)
; I: -
; O: wenn X < Y: A = FFH, CY = 1, Z = 0, S = 1
;    wenn X = Y: A = 00H, CY = 0, Z = 1, S = 0
;    wenn X > Y: A = 01H, CY = 0, Z = 0, S = 0

  0A78 CD4F0A          CALL    0A4FH          :Vergleiche X mit Y
  0A7B C25E09          JP      NZ,095EH       :Nach TEST2 wenn X <> Y
  0A7E C9              RET

; X = HL = CINT ( X )

  0A7F E7              RST     20H            :TSTTYP
  0A80 2A2141          LD      HL,(4121H)     :HL = X (INT)
  0A83 F8              RET     M              :Fertig wenn X bereits im
                                              :INT-Format
  0A84 CAF60A          JP      Z,0AF6H        :TM-Error, wenn X im STR-Format
  0A87 D4B90A          CALL    NC,0AB9H       :CSNG, wenn X im DBL-Format
  0A8A 21B207          LD      HL,07B2H       :OV-Error als RET-Adr setzen
  0A8D E5              PUSH    HL
  0A8E 3A2441          LD      A,(4124H)      :A = Exp (X)
  0A91 FE90            CP      90H            :Exp > 2 hoch 16 ? (16 Bit)
  0A93 300E            JR      NC,0AA3H       :Ja: Dann X genau -32768 ?
  0A95 CDFB0A          CALL    0AFBH          :Nein: DE = INT-Wert von X
  0A98 EB              EX      DE,HL          :HL = INT-Wert
  0A99 D1              POP     DE             :RET-Adresse löschen (07B2H)

; X = HL (INT) und VT = INT setzen

  0A9A 222141          LD      (4121H),HL     :INT-Wert nach X

; VT auf INT setzen

  0A9D 3E02            LD      A,02H          :A = 02H
  0A9F 32AF40          LD      (40AFH),A      :VT setzen
  0AA2 C9              RET

; Fortsetzung von CINT(X)
; Ist X genau -32768 ? (also noch im INT-Format ?)

  0AA3 018090          LD      BC,9080H       :BCDE = -32768
  0AA6 110000          LD      DE,0000H
  0AA9 CD0C0A          CALL    0A0CH          :X und BCDE vergleichen
  0AAC C0              RET     NZ             :RET (OV-Error), wenn ungleich
  0AAD 61              LD      H,C            :HL = -32768
  0AAE 6A              LD      L,D
  0AAF 18E8            JR      0A99H          :RET-Adresse löschen und
                                              :X = HL setzen
```

```
; X = CSNG ( X )

  0AB1 E7            RST    20H          :TSTTYP
  0AB2 E0            RET    PO           ;Fertig wenn X bereits im
                                         ;INT-Format
  0AB3 FACC0A        JP     M,0ACCH      ;weiter bei 0ACCH, wenn X im
                                         ;INT-Format
  0AB6 CAF60A        JP     Z,0AF6H      ;TM-Error, wenn X im STR-Format
  0AB9 CDBF09        CALL   09BFH        ;BCDE = X (SNG) (Die vier nie-
                                         ;drigsten LSBs eines DBL-Wertes
                                         ;werden weggelassen
  0ABC CDEF0A        CALL   0AEFH        ;VT auf SNG setzen
  0ABF 78            LD     A,B          ;BCDE = 0 ?
  0AC0 B7            OR     A
  0AC1 C8            RET    Z            ;Ja: ok
  0AC2 CDDF09        CALL   09DFH        ;Nein: Mantissen berichtigen.
                                         ;Signs ausblenden
  0AC5 212041        LD     HL,4120H
  0AC8 46            LD     B,(HL)       ;B = 3. LSB (X) bei DBL
  0AC9 C39607        JP     0796H        ;X = BCDE und runden

; X = CSNG ( X ) (INT)

  0ACC 2A2141        LD     HL,(4121H)   ;HL = INT-Wert

; X = CSNG ( HL ) (INT)

  0ACF CDEF0A        CALL   0AEFH        ;VT auf SNG setzen
  0AD2 7C            LD     A,H          ;A = MSB (HL)
  0AD3 55            LD     D,L          ;D = LSB (HL)
  0AD4 1E00          LD     E,00H        ;E = 00H
  0AD6 0690          LD     B,90H        ;B = 90H (Exp für 2 hoch 16)
  0AD8 C36909        JP     0969H        ;Nach FLOAT springen

; X = CDBL ( X )

  0ADB E7            RST    20H          :TSTTYP
  0ADC D0            RET    NC           ;Fertig wenn X bereits im
                                         ;DBL-Format
  0ADD CAF60A        JP     Z,0AF6H      ;TM-Error, wenn X im STR-Format
  0AE0 FCCC0A        CALL   M,0ACCH      ;CSNG, wenn X im INT-Format
  0AE3 210000        LD     HL,0000H     ;LSBs einfach auf 00H setzen
  0AE6 221D41        LD     (411DH),HL
  0AE9 221F41        LD     (411FH),HL

; VT auf DBL setzen

  0AEC 3E08          LD     A,08H        ;A = 08H
  0AEE 013E04        LD     BC,043EH     ;--

; VT auf SNG setzen

 *0AEF   3E04        LD     A,04H        ;A = 04H
  0AF1 C39F0A        JP     0A9FH        ;VT = A
```

```
; Test ob X im STR-Format sonst TM-Error

  0AF4 E7              RST     20H          :TSTTYP
  0AF5 C8              RET     Z            :X im STR-Format ?
                                            :Ja: RET

; TM - Error

  0AF6 1E18            LD      E,18H        :E = Fehlercode
  0AF8 C3A219          JP      19A2H        :Zur Fehlerroutine springen

; UPRO für INT, FIX, CINT
; I: X  = SNG-Wert mit Exp <= 98H (2 hoch 24)
        (also mit Nachkommastellen)
;    A  = Exp (X)
; O: DE = INT-Wert von X

  0AFB 47              LD      B,A          :BCDE = A
  0AFC 4F              LD      C,A
  0AFD 57              LD      D,A
  0AFE 5F              LD      E,A
  0AFF B7              OR      A            :A = 0 ?
  0B00 C8              RET     Z            :Ja: RET mit DE = 0000H
  0B01 E5              PUSH    HL           :HL retten
  0B02 CDBF09          CALL    09BFH        :BCDE = X
  0B05 CDDF09          CALL    09DFH        :Mantisse und Sign behandeln
                                            :(Von X und BCDE, Signs immer
                                            :gleich, da BCDE = X)
  0B08 AE              XOR     (HL)         :A,7 = Sign von BCDE bzw X
  0B09 67              LD      H,A          :H,7 = Sign (X)
  0B0A FC1F0B          CALL    M,0B1FH      :Ist X negativ ?
                                            :Ja: BCDE abrunden
  0B0D 3E98            LD      A,98H        :A = Exp für 2 hoch 24
                                            :(24 Bit Mantisse, keine
                                            :Nachkommastellen)
  0B0F 90              SUB     B            :A = 98H - Exp (X)
  0B10 CDD707          CALL    07D7H        :CDE um A Bits nach rechts
                                            :schieben (B wird LSB)
                                            :Alle Nachkommastellen
                                            :rausschieben
  0B13 7C              LD      A,H          :A,7 = Sign (X)
  0B14 17              RLA                  :CY = Sign
  0B15 DCA807          CALL    C,07A8H      :CDE aufrunden, wenn X negativ
  0B18 0600            LD      B,00H        :LSB löschen
  0B1A DCC307          CALL    C,07C3H      :CDEB invertieren, wenn X negativ
  0B1D E1              POP     HL           :HL zurück
  0B1E C9              RET
```

```
; BCDE abrunden

  0B1F 1B              DEC    DE          ;LSBs abrunden
  0B20 7A              LD     A,D         ;LSBs prüfen
  0B21 A3              AND    E           ;War DE = 0000H ?
  0B22 3C              INC    A           ;Ja: Dann ist A jetzt 00H
  0B23 C0              RET    NZ          ;Fertig wenn LSBs <> 0 waren
  0B24 0B              DEC    BC          ;Sonst MSBs abrunden
  0B25 C9              RET

; X = FIX ( X )

  0B26 E7              RST    20H         ;TSTTYP
  0B27 F8              RET    M           ;Fertig wenn X bereits im
                                          ;INT-Format
  0B28 CD5509          CALL   0955H       ;TEST2
  0B2B F2370B          JP     P,0B37H     ;INT(X) ausführen, wenn X >= 0
  0B2E CD8209          CALL   0982H       ;X = -X
  0B31 CD370B          CALL   0B37H       ;INT(X) ausführen
  0B34 C37B09          JP     097BH       ;X = ABS (X) und RET

; X = INT ( X )

  0B37 E7              RST    20H         ;TSTTYP
  0B38 F8              RET    M           ;Fertig wenn X bereits im
                                          ;INT-Format
  0B39 301E            JR     NC,0B59H    ;weiter bei 0B59H wenn X im
                                          ;DBL-Format
  0B3B 28B9            JR     Z,0AF6H     ;TM-Error, wenn X im STR-Format

; X = INT ( X ) (SNG)
; I: X = SNG-Wert
; O: X = SNG-Wert ohne Nachkommastellen
;    A = LSB des INT-Wertes von X

  0B3D CD8E0A          CALL   0A8EH       ;X = CINT (X) (SNG)
  0B40 212441          LD     HL,4124H    ;HL : Exp (X)
  0B43 7E              LD     A,(HL)      ;A = Exp (X)
  0B44 FE98            CP     98H         ;Exp >= 2 hoch 24 ?
                                          ;Ja: Dann hat X keine
                                          ;Nachkommastellen
  0B46 3A2141          LD     A,(4121H)   ;A = LSB
  0B49 D0              RET    NC          ;Fertig wenn Exp (X) >= 98H
  0B4A 7E              LD     A,(HL)      ;A = Exp (X)
  0B4B CDFB0A          CALL   0AFBH       ;Nachkommastellen entfernen
  0B4E 3698            LD     (HL),98H    ;Exp auf 2 hoch 24 setzen
  0B50 7B              LD     A,E         ;A = LSB
  0B51 F5              PUSH   AF          ;LSB retten
  0B52 79              LD     A,C         ;A = MSB
  0B53 17              RLA                ;Sign nach CY
  0B54 CD6207          CALL   0762H       ;FLOAT aufrufen
  0B57 F1              POP    AF          ;LSB zurück nach A
  0B58 C9              RET
```

; X = INT (X) (DBL)

```
0B59 212441         LD      HL,4124H        :HL : Exp (X)
0B5C 7E             LD      A,(HL)          :A = Exp (X)
0B5D FE90           CP      90H             :Exp (X) < 2 hoch 16 ?
0B5F DA7F0A         JP      C,0A7FH         :Ja: CINT (X) ausführen
0B62 2014           JR      NZ,0B78H        :Sprung wenn Exp <> 2 hoch 16
0B64 4F             LD      C,A             :C = 90H (Exp ist 2 hoch 16 !)
0B65 2B             DEC     HL              :HL = MSB(X)
0B66 7E             LD      A,(HL)          :A = MSB(X)
0B67 EE80           XOR     80H             :A = 00H, wenn X = -32768
0B69 0606           LD      B,06H           :Die übrigen 6 Bytes testen
0B6B 2B             DEC     HL              :Zeiger -1
0B6C B6             OR      (HL)            :Byte testen
0B6D 05             DEC     B               :Zähler -1
0B6E 20FB           JR      NZ,0B6BH        :Nächstes Byte
0B70 B7             OR      A               :Waren alle Bytes = 00H ?
0B71 210080         LD      HL,8000H        :HL = -32768 (INT-Wert)
0B74 CA9A0A         JP      Z,0A9AH         :Ja: X war - 32768
0B77 79             LD      A,C             :A = Exp (X) = 90H (wegen 0B64H)
0B78 FEB8           CP      0B8H            :Exp >= 2 hoch 56 ?
                                            :(56 Bit Mantisse)
0B7A D0             RET     NC              :Ja: Keine Nachkommastellen
                                            :vorhanden
0B7B F5             PUSH    AF              :Exp retten
0B7C CDBF09         CALL    09BFH           :BCDE = X (SNG)
0B7F CDDF09         CALL    09DFH           :Mantisse berichtigen
0B82 AE             XOR     (HL)            :A,7 = Sign von X bzw BCDE
0B83 2B             DEC     HL              :HL : Exp (X)
0B84 36B8           LD      (HL),0B8H       :Exp auf 2 hoch 56 setzen
0B86 F5             PUSH    AF              :Sign retten
0B87 FCA00B         CALL    M,0BA0H         :X abrunden, wenn X negativ
0B8A 212341         LD      HL,4123H        :HL = MSB (X)
0B8D 3EB8           LD      A,0B8H          :A = Exp für 2 hoch 56
0B8F 90             SUB     B               :Exp-Differenz ausrechnen
0B90 CD690D         CALL    0D69H           :(HL) - (HL-7) (entspricht X)
                                            :um A Bits nach rechts schieben
                                            :-> Nachkommastellen werden
                                            :gelöscht
0B93 F1             POP     AF              :Sign zurück
0B94 FC200D         CALL    M,0D20H         :Mantisse von X aufrunden,
                                            :wenn X negativ war
0B97 AF             XOR     A               :A = 00H
0B98 321C41         LD      (411CH),A       :LSB-Unterlauf von X löschen
0B9B F1             POP     AF              :Exp zurück nach A
0B9C D0             RET     NC              :Fertig wenn CALL von 12B8H kam
                                            :(sonst ist CY = 1 wegen 0B7AH)
0B9D C3D80C         JP      0CD8H           :weiter bei DFLOAT
```

```
: Mantisse von X (DBL) abrunden

  0BA0 211D41          LD      HL,411DH      :HL = Zeiger auf X
  0BA3 7E              LD      A,(HL)        :A = Byte
  0BA4 35              DEC     (HL)          :Abrunden
  0BA5 B7              OR      A             :War Byte = 00H ?
  0BA6 23              INC     HL            :Zeiger +1
  0BA7 28FA            JR      Z,0BA3H       :Ja: Nächstes Byte
  0BA9 C9              RET                   :Nein: Fertig

: UPRO zur Feldgrößenberechnung bei DIM
: DE = DE * BC

  0BAA E5              PUSH    HL            :PTZ retten
  0BAB 210000          LD      HL,0000H      :Ergebnis = 0000H
  0BAE 78              LD      A,B -         :BC = 0 ?
  0BAF B1              OR      C
  0BB0 2812            JR      Z,0BC4H       :Ja: RET mit Ergebnis 0
  0BB2 3E10            LD      A,10H         :A = Zähler für 16 Bit
  0BB4 29              ADD     HL,HL         :Ergebnis links schieben
  0BB5 DA3D27          JP      C,273DH       :BS-Error, wenn Überlauf
  0BB8 EB              EX      DE,HL         :HL retten
  0BB9 29              ADD     HL,HL         :Höchstes Bit von DE nach CY
  0BBA EB              EX      DE,HL         :HL zurück
  0BBB 3004            JR      NC,0BC1H      :Sprung, wenn Bit = 0
  0BBD 09              ADD     HL,BC         :Sonst BC zum Erg. addieren
  0BBE DA3D27          JP      C,273DH       :BS-Error, wenn Überlauf
  0BC1 3D              DEC     A             :Zähler -1
  0BC2 20F0            JR      NZ,0BB4H      :Nächstes Bit
  0BC4 EB              EX      DE,HL         :DE = Ergebnis
  0BC5 E1              POP     HL            :PTZ zurück
  0BC6 C9              RET

: ISUB: X = HL = DE - HL (INT)
: I: DE = Minuend
:    HL = Subtrahend
: O: HL = Differenz falls im INT-Bereich
     X  = Differenz (Automatische Umwandlung in SNG-Format, falls Ergebnis
          nicht im INT-Bereich)

  0BC7 7C              LD      A,H           :A = MSB von HL
  0BC8 17              RLA                   :CY = Sign von HL
  0BC9 9F              SBC     A,A           :A = 00H, wenn HL >= 0
                                             :A = FFH, wenn HL <  0
  0BCA 47              LD      B,A           :B = A
  0BCB CD510C          CALL    0C51H         :HL = -HL, CY = 1
  0BCE 79              LD      A,C           :A = 00H (C = 00H von 0C51H)
  0BCF 98              SBC     A,B           :A = FFH, wenn HL >= 0 war
                                             :    (HL ist jetzt < 0)
                                             :A = 00H, wenn HL <  0 war
                                             :    (HL ist jetzt >= 0)
  0BD0 1803            JR      0BD5H         :Nach IADD springen
                                             :(Also DE = DE + (-HL) ausführen)
```

```
; IADD: X = HL = DE + HL (INT)
; I: DE = 1. Summand
;     HL = 2. Summand
; O: HL = Summe falls im INT-Bereich
      X  = Summe (Automatische Umwandlung ins SNG-Format, falls Ergebnis
;            nicht im INT-Bereich)

  0BD2 7C             LD     A,H          ;Vorzeichenflag berechnen
  0BD3 17             RLA                 ;(siehe ISUB)
  0BD4 9F             SBC    A,A
  0BD5 47             LD     B,A          ;B = 00H, wenn HL >= 0
                                          ;B = FFH, wenn HL <  0
  0BD6 E5             PUSH   HL           ;2. Summanden retten
  0BD7 7A             LD     A,D          ;Vorzeichenflag des 2. Summanden
  0BD8 17             RLA                 ;berechnen
  0BD9 9F             SBC    A,A
  0BDA 19             ADD    HL,DE        ;Addition ausführen
  0BDB 88             ADC    A,B          ;Vorzeichenflags und Überlauf
  0BDC 0F             RRCA                ;verrechnen:
  0BDD AC             XOR    H            ;A.7 ist gesetzt, wenn bei
                                          ;gleichem Vorzeichen der beiden
                                          ;Summanden, ein Überlauf erfolgt
                                          ;ist oder bei verschiedenem
                                          ;Vorzeichen kein Überlauf
                                          ;erfolgte
  0BDE F2990A         JP     P,0A99H      ;A.7 = 0 ? Ja: Ergebnis ist ok
  0BE1 C5             PUSH   BC           ;Signflag des 2. Summanden retten
  0BE2 EB             EX     DE,HL        ;HL = 1. Summand
  0BE3 CDCF0A         CALL   0ACFH        ;X = CSNG (HL)
  0BE6 F1             POP    AF           ;Signflag zurück
  0BE7 E1             POP    HL           ;2. Summand zurück
  0BE8 CDA409         CALL   09A4H        ;(SP) = X = 1. Summand
  0BEB EB             EX     DE,HL        ;DE = 2. Summand
  0BEC CD6B0C         CALL   0C6BH        ;X = SFLOAT (DE)
  0BEF C38F0F         JP     0F8FH        ;X = X + (SP) (Addition im
                                          ;SNG-Format)
```

```
; IMUL: X = HL = DE * HL (INT)
; I: DE = Multiplikant
;    HL = Multiplikator (beide Werte im INT-Format)
; O: HL = Produkt falls im INT-Bereich
;    X  = Produkt (Automatische Umwandlung ins SNG-Format, falls Ergebnis
;         nicht im INT-Bereich)

0BF2 7C         LD     A,H          ;HL = 0 ?
0BF3 B5         OR     L
0BF4 CA9A0A     JP     Z,0A9AH      ;Ja: Ergebnis = 0
0BF7 E5         PUSH   HL           ;Multiplikator retten
0BF8 D5         PUSH   DE           ;Multiplikant retten
0BF9 CD450C     CALL   0C45H        ;Sign ausblenden, Beide Zahlen
                                    ;positiv machen
0BFC C5         PUSH   BC           ;Signflag retten (B.7 = 0 bei
                                    ;gleichen Vorzeichen)
0BFD 44         LD     B,H          ;BC = HL
0BFE 4D         LD     C,L
0BFF 210000     LD     HL,0000H     ;Ergebnis = 0
0C02 3E10       LD     A,10H        ;A = Zähler für 16 Bits
0C04 29         ADD    HL,HL        ;Ergebnis links schieben
0C05 381F       JR     C,0C26H      ;Überlauf ? Ja: weiter bei 0C26H
0C07 EB         EX     DE,HL        ;Nächstes Bit von DE nach CY
0C08 29         ADD    HL,HL
0C09 EB         EX     DE,HL
0C0A 3004       JR     NC,0C10H     ;Sprung wenn Bit nicht gesetzt
0C0C 09         ADD    HL,BC        ;BC zum Ergebnis addieren, wenn
                                    ;Bit in DE gesetzt war
0C0D DA260C     JP     C,0C26H      ;Sprung bei Überlauf
0C10 3D         DEC    A            ;Zähler -1
0C11 20F1       JR     NZ,0C04H     ;Nächstes Bit
0C13 C1         POP    BC           ;Signflag zurück
0C14 D1         POP    DE           ;Multiplikant zurück
0C15 7C         LD     A,H          ;Ist das Ergebnis negativ ?
0C16 B7         OR     A            ;bzw. Ist ein Überlauf in das
                                    ;Vorzeichenbit erfolgt ? (>32767)
0C17 FA1F0C     JP     M,0C1FH      ;Ja: weiter bei 0C1FH
0C1A D1         POP    DE           ;Multiplikator zurück
0C1B 78         LD     A,B          ;A = Signflag
0C1C C34D0C     JP     0C4DH        ;Sign des Ergebnisses setzen

; Überlauf ins 15. Bit (Vorzeichenbit)
; Da beide Faktoren positiv gemacht wurden, muß das Ergebnis auch positiv sein,
; d.h. das 16. Bit muß 0 sein

0C1F EE80       XOR    80H          ;A = 00H, wenn H = 80H
0C21 B5         OR     L            ;Ist HL = 8000H = 32768 ?
                                    ;(ohne Vorzeichen)
0C22 2813       JR     Z,0C37H      ;Ja: weiter bei 0C37H
0C24 EB         EX     DE,HL        ;Nein: HL = Multiplikant
0C25 01C1E1     LD     BC,0E1C1H    ;--
```

```
; Überlauf bei IMUL
; Beide Faktoren in SNG-Format umwandeln und dann SMUL ausführen

*0C26    C1              POP    BC          :B = Signflag
*0C27       E1           POP    HL          :HL = Multiplikant
 0C28 CDCF0A             CALL   0ACFH       :X = CSNG (HL)
 0C2B E1                 POP    HL          :HL = Multiplikator
 0C2C CDA409             CALL   09A4H       :(SP) = X = Multiplikant
 0C2F CDCF0A             CALL   0ACFH       :X = CSNG (HL) = Multiplikator
 0C32 C1                 POP    BC          :BCDE = (SP) = Multiplikant
 0C33 D1                 POP    DE
 0C34 C34708             JP     0847H       :X = BCDE * X (SNG)

; Das Ergebnis ist 32768 (ohne Vorzeichen)

 0C37 78                 LD     A,B         :A = Signflag
 0C38 B7                 OR     A           :Ungleiche Vorzeichen ?
 0C39 C1                 POP    BC          :Stack korrigieren
 0C3A FA9A0A             JP     M,0A9AH     :Ja: Ergebnis = 8000H = -32768
                                            :Nein: Das Ergebnis ist +32768
 0C3D D5                 PUSH   DE          :DE retten
 0C3E CDCF0A             CALL   0ACFH       :X = CSNG (HL) = -32768
 0C41 D1                 POP    DE          :DE zurück
 0C42 C38209             JP     0982H       :X = -X (Ergebnis = +32768)

; Vorzeichenprüfung bei IMUL:
; Bei DE und HL das Vorzeichen ausblenden (positiv machen)
; I: DE  = Multiplikant
;    HL  = Multiplikator (beide Werte im INT-Format)
; O: DE  = ABS (Multiplikator)
;    HL  = ABS (Multiplikant)
;    B.7 = 0 bei gleichen Vorzeichen, sonst B.7 = 1

 0C45 7C                 LD     A,H         :Vorzeichen von HL
 0C46 AA                 XOR    D           :und DE verknüpfen
 0C47 47                 LD     B,A         :B.7 = 0 bei gleichen Vorzeichen
 0C48 CD4C0C             CALL   0C4CH       :HL = ABS (HL)
 0C4B EB                 EX     DE,HL       :desgl. mit DE
 0C4C 7C                 LD     A,H         :HL testen
 0C4D B7                 OR     A
 0C4E F29A0A             JP     P,0A9AH     :OK, wenn HL >= 0

; X = HL = -HL (INT)
; I: HL = INT-Wert
; O: HL = Negativer INT-Wert
;    X  = HL und VT = INT

 0C51 AF                 XOR    A           :A = 00H
 0C52 4F                 LD     C,A         :C = 00H
 0C53 95                 SUB    L
 0C54 6F                 LD     L,A         :L = 00H - L
 0C55 79                 LD     A,C         :A = 00H
 0C56 9C                 SBC    A,H
 0C57 67                 LD     H,A         :H = 00H - H - CY
 0C58 C39A0A             JP     0A9AH       :HL nach X und VT = INT
```

```
; X = -X (INT)
; Umwandlung in SNG, falls X = -32768
; (denn +32768 ist nicht mehr im INT-Bereich)

  0C5B 2A2141        LD     HL,(4121H)  ;HL = X
  0C5E CD510C        CALL   0C51H       ;X = HL = -HL
  0C61 7C            LD     A,H         ;War HL = -32768 ?
  0C62 EE80          XOR    80H
  0C64 B5            OR     L
  0C65 C0            RET    NZ          ;Nein: Wert ok
  0C66 EB            EX     DE,HL       ;DE = Wert
  0C67 CDEF0A        CALL   0AEFH       ;VT = SNG
  0C6A AF            XOR    A           ;A = 00H
  0C6B 0698          LD     B,98H       ;B = Exp für 2 hoch 24
  0C6D C36909        JP     0969H       ;Nach FLOAT springen

; X = X - Y = X + (-Y) (DBL)

  0C70 212D41        LD     HL,412DH    ;HL : MSB (Y)
  0C73 7E            LD     A,(HL)      ;Y = -Y
  0C74 EE80          XOR    80H
  0C76 77            LD     (HL),A

; X = X + Y (DBL)

  0C77 212E41        LD     HL,412EH    ;HL : Exp (Y)
  0C7A 7E            LD     A,(HL)      ;A = Exp (Y)
  0C7B B7            OR     A           ;Y = 0 ?  (Exp (Y) = 0 ?)
  0C7C C8            RET    Z           ;Ja: X ist Ergebnis
  0C7D 47            LD     B,A         ;B = Exp (Y)
  0C7E 2B            DEC    HL          ;HL : MSB (Y)
  0C7F 4E            LD     C,(HL)      ;C = MSB (Y)
  0C80 112441        LD     DE,4124H    ;DE : Exp (X)
  0C83 1A            LD     A,(DE)      ;A = Exp (X)
  0C84 B7            OR     A           ;X = 0 ? (Exp (X) = 0 ?)
  0C85 CAF409        JP     Z,09F4H     ;Ja: X = Y (Y ist Ergebnis)
  0C88 90            SUB    B           ;A = Exp (X) - Exp (Y)
  0C89 3016          JR     NC,0CA1H    ;Sprung, wenn Exp (X) >= Exp (Y)
                                        ;Sonst X und Y vertauschen
  0C8B 2F            CPL                ;A = -A
  0C8C 3C            INC    A           ;A +1 für 2er Komplement
  0C8D F5            PUSH   AF          ;Exp-Differenz retten
  0C8E 0E08          LD     C,08H       ;8 Bytes vertauschen
  0C90 23            INC    HL          ;HL auf Exp (Y) zeigen lassen
  0C91 E5            PUSH   HL          ;Zeiger retten
  0C92 1A            LD     A,(DE)      ;Bytes von (DE)
  0C93 46            LD     B,(HL)      ;und (HL)
  0C94 77            LD     (HL),A      ;vertauschen
  0C95 78            LD     A,B
  0C96 12            LD     (DE),A
```

```
0C97 1B         DEC    DE           :Zeiger -1
0C98 2B         DEC    HL
0C99 0D         DEC    C            :Zähler -1
0C9A 20F6       JR     NZ,0C92H     :Nächstes Byte
0C9C E1         POP    HL           :Zeiger auf Exp (X) zurück
0C9D 46         LD     B,(HL)       :B = Exp (X)
0C9E 2B         DEC    HL
0C9F 4E         LD     C,(HL)       :C = MSB (X)
0CA0 F1         POP    AF           :Exp-Differenz zurück
                                    :In X ist jetzt der größere
                                    :Summand
0CA1 FE39       CP     39H          :Exp-Diff. größer als 2 hoch 56 ?
                                    :(56 Bits DBL-Mantisse)
0CA3 D0         RET    NC           :Ja: Y ist zu klein, die Summe
                                    :würde X nicht verändern
0CA4 F5         PUSH   AF -         :Exp-Differenz retten
0CA5 CDDF09     CALL   09DFH        :Mantissen berichtigen, Vor-
                                    :zeichen verrechnen
0CA8 23         INC    HL           :HL : Exp (X)
0CA9 3600       LD     (HL),00H     :Exp auf 0 setzen
0CAB 47         LD     B,A          :Signflag nach B retten
0CAC F1         POP    AF           :Exp-Differenz zurück
0CAD 212D41     LD     HL,412DH     :HL = Zeiger auf Y
0CB0 CD690D     CALL   0D69H        :(HL) bis (HL-7) um A Bits nach
                                    :rechts schieben (X und Y auf
                                    :gleichen Exponenten bringen)
0CB3 3A2641     LD     A,(4126H)    :Unterlauf nach
0CB6 321C41     LD     (411CH),A    :X kopieren
0CB9 78         LD     A,B          :A = Signflag
0CBA B7         OR     A            :Waren die Vorzeichen gleich ?
0CBB F2CF0C     JP     P,0CCFH      :Nein: weiter bei 0CCFH
0CBE CD330D     CALL   0D33H        :Ja: Mantissen addieren
0CC1 D20E0D     JP     NC,0D0EH     :Sprung, wenn kein Überlauf
0CC4 EB         EX     DE,HL        :Überlauf: HL : Exp (X)
0CC5 34         INC    (HL)         :Exp (X) +1
0CC6 CAB207     JP     Z,07B2H      :OV-Error, wenn Exp-Überlauf
0CC9 CD900D     CALL   0D90H        :X um 1 Bit nach rechts schieben
                                    :(X = X/2, da durch Exp +1
                                    :X mit 2 multipliziert wurde)
0CCC C30E0D     JP     0D0EH        :weiter bei 0D0EH

; Ungleiche Vorzeichen: Mantissen subtrahieren

0CCF CD450D     CALL   0D45H        :Mantissen subtrahieren
0CD2 212541     LD     HL,4125H     :HL : Signflag
0CD5 DC570D     CALL   C,0D57H      :Mantisse von X bei Unterlauf
                                    :negieren
```

```
: DFLOAT (DBL)
: DBL-Mantisse solange nach links schieben, bis das höchste Bit der Mantisse
: gleich 1 und der Exponent möglichst klein ist

   0CD8 AF               XOR    A            ;A = 00H
   0CD9 47               LD     B,A          ;B = Verschiebungszähler
   0CDA 3A2341           LD     A,(4123H)    ;A = MSB (X)
   0CDD B7               OR     A            ;MSB = 0 ?
   0CDE 201E             JR     NZ,0CFEH     ;Nein: Bits des MSBs testen
                                             ;Ja: X um 1 Byte nach links
                                             ;schieben:
   0CE0 211C41           LD     HL,411CH     ;HL = Zeiger auf Unterlauf von X
   0CE3 0E08             LD     C,08H        ;C = Zähler für 8 Bytes
   0CE5 56               LD     D,(HL)       ;Neues Byte holen
   0CE6 77               LD     (HL),A       ;Altes Byte einsetzen
   0CE7 7A               LD     A,D -        ;Altes Byte = Neues Byte
   0CE8 23               INC    HL           ;Zeiger +1
   0CE9 0D               DEC    C            ;Zähler -1
   0CEA 20F9             JR     NZ,0CE5H     ;Nächstes Byte
   0CEC 78               LD     A,B          ;A = Verschiebungszähler
   0CED D608             SUB    08H          ;8 (für 8 Bits) abziehen
   0CEF FEC0             CP     0C0H         ;schon -64 erreicht ?
                                             ;(schon 8 Bytes verschoben ?)
   0CF1 20E6             JR     NZ,0CD9H     ;Nein: MSB neu testen
   0CF3 C37807           JP     0778H        ;Ja: Alle Bytes von X waren 00H
                                             ;dann X = 0 setzen

: Bitweise weiterschieben

   0CF6 05               DEC    B            ;Verschiebungszähler -1
   0CF7 211C41           LD     HL,411CH     ;HL : Unterlauf von X
   0CFA CD970D           CALL   0D97H        ;(HL) bis (HL+7) (also X) um
                                             ;1 Bit nach links schieben
   0CFD B7               OR     A            ;A = Neues MSB, A,7 = 1 ?
   0CFE F2F60C           JP     P,0CF6H      ;Nein: weiter verschieben
   0D01 78               LD     A,B          ;A = Verschiebungszähler
   0D02 B7               OR     A            ;Nichts verschoben ?
   0D03 2809             JR     Z,0D0EH      ;Ja: X fertig
   0D05 212441           LD     HL,4124H     ;Nein: Von Exp die Anzahl der
   0D08 86               ADD    A,(HL)       ;verschobenen Bits abziehen
   0D09 77               LD     (HL),A       ;(Eine Verschiebung ist gleich-
                                             ;bedeutend mit X = X * 2)
   0D0A D27807           JP     NC,0778H     ;X gleich 0 setzen wenn kein
                                             ;Überlauf entstand (Der Verschie-
                                             ;bungszähler ist negativ !)
   0D0D C8               RET    Z            ;Fertig wenn Exp = 0
   0D0E 3A1C41           LD     A,(411CH)    ;A = Unterlauf Byte
   0D11 B7               OR     A            ;A.7 = 1 ?
   0D12 FC200D           CALL   M,0D20H      ;Ja: X aufrunden
   0D15 212541           LD     HL,4125H     ;HL : Signflag
   0D18 7E               LD     A,(HL)       ;A = Signflag
   0D19 E680             AND    80H          ;Sign maskieren, A,7 = 1 bei
                                             ;gleichen Vorzeichen
```

```
0D1B 2B              DEC     HL
0D1C 2B              DEC     HL              ;HL : MSB (X)
0D1D AE              XOR     (HL)            ;Signflag mit MSB (X) verknüpfen
0D1E 77              LD      (HL),A          ;Neuen MSB incl. Sign setzen
0D1F C9              RET

; X (DBL) aufrunden

0D20 211D41          LD      HL,411DH        ;HL : LSB (X)
0D23 0607            LD      B,07H           ;7 Bytes Mantisse
0D25 34              INC     (HL)            ;Byte +1, Überlauf ?
0D26 C0              RET     NZ              ;Nein: Fertig
0D27 23              INC     HL              ;Ja: Zeiger +1
0D28 05              DEC     B               ;Zähler +1
0D29 20FA            JR      NZ,0D25H        ;Nächstes Byte aufrunden
0D2B 34              INC     (HL)-           ;Bei Überlauf vom MSB: Exp +1
0D2C CAB207          JP      Z,07B2H         ;OV-Error bei Überlauf des Exp
0D2F 2B              DEC     HL              ;Sonst MSB.7 auf 1 setzen
0D30 3680            LD      (HL),80H
0D32 C9              RET

; Mantissen von X und Y addieren

0D33 212741          LD      HL,4127H        ;HL : LSB (Y)
0D36 111D41          LD      DE,411DH        ;DE : LSB (X)
0D39 0E07            LD      C,07H           ;7 Bytes Mantisse
0D3B AF              XOR     A               ;CY = 0
0D3C 1A              LD      A,(DE)          ;A = Byte von X
0D3D 8E              ADC     A,(HL)          ;Byte von Y aufaddieren
0D3E 12              LD      (DE),A          ;Summe in X ablegen
0D3F 13              INC     DE              ;Zeiger +1
0D40 23              INC     HL
0D41 0D              DEC     C               ;Zähler -1
0D42 20F8            JR      NZ,0D3CH        ;Nächstes Byte
0D44 C9              RET

; Mantissen von X und Y subtrahieren

0D45 212741          LD      HL,4127H        ;HL : LSB (Y)
0D48 111D41          LD      DE,411DH        ;DE : LSB (X)
0D4B 0E07            LD      C,07H           ;7 Bytes Mantisse
0D4D AF              XOR     A               ;CY = 0
0D4E 1A              LD      A,(DE)          ;A = Byte von X
0D4F 9E              SBC     A,(HL)          ;Byte von Y abziehen
0D50 12              LD      (DE),A          ;Differenz in X ablegen
0D51 13              INC     DE              ;Zeiger +1
0D52 23              INC     HL
0D53 0D              DEC     C               ;Zähler -1
0D54 20F8            JR      NZ,0D4EH        ;Nächstes Byte
0D56 C9              RET
```

```
; Mantisse von X (incl. Unterlauf) und Signflag negieren

  0D57 7E         LD      A,(HL)      :A = Signflag
  0D58 2F         CPL                 :Negation
  0D59 77         LD      (HL),A      :Signflag zurückschreiben
  0D5A 211C41     LD      HL,411CH    :HL : Unterlauf von X
  0D5D 0608       LD      B,08H       :8 Bytes negieren
  0D5F AF         XOR     A           :A = 00H
  0D60 4F         LD      C,A         :C = 00H
  0D61 79         LD      A,C         :A = 00H
  0D62 9E         SBC     A,(HL)      :A = 00H - (HL) - CY
  0D63 77         LD      (HL),A      :Differenz zurückschreiben
  0D64 23         INC     HL          :Zeiger +1
  0D65 05         DEC     B           :Zähler -1
  0D66 20F9       JR      NZ,0D61H    :Nächstes Byte
  0D68 C9         RET     -

; (HL) bis (HL-7) um A Bits nach rechts schieben

  0D69 71         LD      (HL),C      :MSB abspeichern
  0D6A E5         PUSH    HL          :Zeiger retten
  0D6B D608       SUB     08H         :Mehr als 8 Bits zu verschieben ?
  0D6D 380E       JR      C,0D7DH     :Nein: Bits schieben bei 0D7DH
                                      :Ja: Bytes schieben
  0D6F E1         POP     HL          :Zeiger zurück
  0D70 E5         PUSH    HL          :Zeiger retten
  0D71 110008     LD      DE,0800H    :8 Bytes Mantisse (mit Unterlauf)
                                      :mit 00H füllen
  0D74 4E         LD      C,(HL)      :C = Neues Byte
  0D75 73         LD      (HL),E      :(HL) = Altes Byte
  0D76 59         LD      E,C         :Altes Byte = Neues Byte
  0D77 2B         DEC     HL          :Zeiger -1
  0D78 15         DEC     D           :Zähler -1
  0D79 20F9       JR      NZ,0D74H    :Nächstes Byte verschieben
  0D7B 18EE       JR      0D6BH       :Noch mehr verschieben ?

; Bitweise schieben

  0D7D C609       ADD     A,09H       :SUB 08H rückgängig machen
  0D7F 57         LD      D,A         :D = Zähler
  0D80 AF         XOR     A           :A = 00H
  0D81 E1         POP     HL          :Zeiger zurück
  0D82 15         DEC     D           :Zähler -1
  0D83 C8         RET     Z           :RET wenn fertig
  0D84 E5         PUSH    HL          :Zeiger retten
  0D85 1E08       LD      E,08H       :8 Bytes Mantisse (mit Unterlauf)
  0D87 7E         LD      A,(HL)      :A = Byte
  0D88 1F         RRA                 :rechts schieben (incl. CY)
  0D89 77         LD      (HL),A      :Byte zurück
  0D8A 2B         DEC     HL          :Zeiger -1
  0D8B 1D         DEC     E           :Zähler -1
  0D8C 20F9       JR      NZ,0D87H    :Nächstes Byte
  0D8E 18F0       JR      0D80H       :Noch mehr verschieben ?
```

```
; Mantisse von X um 1 Bit nach rechts schieben

  0D90 212341          LD     HL,4123H     ;HL = Zeiger auf Mantisse
  0D93 1601            LD     D,01H        ;D = Zähler für 1 Bit
  0D95 18ED            JR     0D84H        ;Weiter bei 0D84H

; (HL) bis (HL+7) um 1 Bit nach links schieben

  0D97 0E08            LD     C,08H        ;8 Bytes Mantisse (mit Unterlauf)
  0D99 7E              LD     A,(HL)       ;A = Byte
  0D9A 17              RLA                 ;rechts schieben (incl. CY)
  0D9B 77              LD     (HL),A       ;Byte zurückschreiben
  0D9C 23              INC    HL           ;Zeiger +1
  0D9D 0D              DEC    C            ;Zähler -1
  0D9E 20F9            JR     NZ,0D99H     ;Nächstes Byte verschieben
  0DA0 C9              RET

; DMUL: X = X * Y (DBL)
; I: X = 1. Faktor (DBL)
;    Y = 2. Faktor (DBL)
; O: X = Produkt

  0DA1 CD5509          CALL   0955H        ;TEST2
  0DA4 C8              RET    Z            ;Ergebnis = 0, wenn X = 0
  0DA5 CD0A09          CALL   090AH        ;Exponenten und Sign verrechnen
  0DA8 CD390E          CALL   0E39H        ;Mantisse des 1. Faktors nach
                                           ;414AH retten und Mantisse von X
                                           ;löschen
  0DAB 71              LD     (HL),C       ;Unterlauf von X auf 0
  0DAC 13              INC    DE           ;DE : LSB des 1. Faktors
  0DAD 0607            LD     B,07H        ;7 Bytes Mantisse verrechnen
  0DAF 1A              LD     A,(DE)       ;A = Byte der Mantisse
  0DB0 13              INC    DE           ;Zeiger +1
  0DB1 B7              OR     A            ;Ist kein Bit gesetzt ?
  0DB2 D5              PUSH   DE           ;Zeiger retten
  0DB3 2817            JR     Z,0DCCH      ;Ja: X um 1 Byte nach rechts
                                           ;schieben und nächstes Byte des
                                           ;1. Faktors holen
  0DB5 0E08            LD     C,08H        ;8 Bits pro Byte
  0DB7 C5              PUSH   BC           ;Zähler retten
  0DB8 1F              RRA                 ;nächstes Bit testen, Bit = 1 ?
  0DB9 47              LD     B,A          ;Byte retten
  0DBA DC330D          CALL   C,0D33H      ;Ja: Mantissen von X und Y
                                           ;addieren
  0DBD CD900D          CALL   0D90H        ;Mantisse von X um 1 Bit nach
                                           ;rechts schieben (nächste Stelle)
  0DC0 78              LD     A,B          ;Byte zurück
  0DC1 C1              POP    BC           ;Zähler zurück
  0DC2 0D              DEC    C            ;Bitzähler -1
  0DC3 20F2            JR     NZ,0DB7H     ;Nächstes Bit testen
  0DC5 D1              POP    DE           ;Zeiger zurück
  0DC6 05              DEC    B            ;Bytezähler -1
  0DC7 20E6            JR     NZ,0DAFH     ;Nächstes Byte testen
  0DC9 C3D80C          JP     0CD8H        ;Sprung nach DFLOAT
```

```
; Mantisse von X um 1 Byte nach rechts schieben

  0DCC 212341        LD      HL,4123H      ;HL = Zeiger auf Mantisse von X
  0DCF CD700D        CALL    0D70H         ;Ein Byte rechts schieben
  0DD2 18F1          JR      0DC5H         ;Zurück nach DMUL

; Konstante 10 (DBL,SNG)

  0DD4 00                                  ;Konstante 10 (DBL)
  0DD5 00
  0DD6 00
  0DD7 00
  0DD8 00                                  ;Konstante 10 (SNG)
  0DD9 00
  0DDA 20
  0DDB 84

; X = X / 10 (DBL)

  0DDC 11D40D        LD      DE,0DD4H      ;DE : Konstante 10 (DBL)
  0DDF 212741        LD      HL,4127H      ;HL = Zeiger auf Y
  0DE2 CDD309        CALL    09D3H         ;Kopiere (DE) nach (HL) (Y = 10)

; DDIV: X = X / Y (DBL)
; I: X = Dividend (DBL)
;    Y = Divisor (DBL)
; O: X = Quotient

  0DE5 3A2E41        LD      A,(412EH)     ;A = Exp (Y)
  0DE8 B7            OR      A             ;Y = 0 ?
  0DE9 CA9A19        JP      Z,199AH       ;Ja: /0-Error
  0DEC CD0709        CALL    0907H         ;Exponenten und Sign verrechnen
  0DEF 34            INC     (HL)          ;Exponent berichtigen
  0DF0 34            INC     (HL)          ;(siehe auch SDIV)
  0DF1 CD390E        CALL    0E39H         ;Dividend retten und Mantisse von
                                           ;X auf 0 setzen
  0DF4 215141        LD      HL,4151H      ;HL = Zeiger auf Unterlaufbyte
                                           ;des Dividenden
  0DF7 71            LD      (HL),C        ;Unterlaufbyte löschen
  0DF8 41            LD      B,C           ;Unterlaufflag auf 0 setzen
  0DF9 114A41        LD      DE,414AH      ;DE : LSB des Dividenden
  0DFC 212741        LD      HL,4127H      ;HL : LSB des Divisors
  0DFF CD4B0D        CALL    0D4BH         ;Mantissen subtrahieren
  0E02 1A            LD      A,(DE)        ;A = Unterlaufbyte
  0E03 99            SBC     A,C           ;A = Unterlaufbyte - CY (C ist 0)
  0E04 3F            CCF                   ;CY invertieren
  0E05 380B          JR      C,0E12H       ;Sprung wenn kein Unterlauf war
                                           ;sonst Subtraktion rückgängig
                                           ;machen
```

```
 0E07 114A41        LD     DE,414AH      ;DE : LSB des Dividenden
 0E0A 212741        LD     HL,4127H      ;HL : LSB des Divisors
 0E0D CD390D        CALL   0D39H         ;Mantissen addieren
 0E10 AF            XOR    A             ;CY = 0
 0E11 DA1204        JP     C,0412H       ;--
*0E12   12          LD     (DE),A        ;Kein Unterlauf: Unterlaufbyte
                                         ;zurückschreiben
*0E13      04       INC    B             ;Unterlaufflag = 1
 0E14 3A2341        LD     A,(4123H)     ;A = MSB (Quotient)
 0E17 3C            INC    A             ;nächstes Bit = 1 ?
 0E18 3D            DEC    A
 0E19 1F            RRA                  ;A,7 zur Rundung nach CY schieben
 0E1A FA110D        JP     M,0D11H       ;Fertig wenn höchstes Bit = 1
 0E1D 17            RLA                  ;A,7 wieder zurückschieben
 0E1E 211D41        LD     HL,411DH      ;HL : LSB (Quotient)
 0E21 0E07          LD     C,07H         ;7 Bytes Mantisse
 0E23 CD990D        CALL   0D99H         ;Quotient um 1 Bit nach links
                                         ;schieben
 0E26 214A41        LD     HL,414AH      ;HL : LSB (Dividend)
 0E29 CD970D        CALL   0D97H         ;Dividend um 1 Bit nach links
                                         ;schieben
 0E2C 78            LD     A,B           ;Unterlaufflag nach A
 0E2D B7            OR     A             ;War ein Unterlauf ?
 0E2E 20C9          JR     NZ,0DF9H      ;Nein: nächstes Bit verrechnen
 0E30 212441        LD     HL,4124H      ;Ja: HL : Exp (Quotient)
 0E33 35            DEC    (HL)          ;Quotient = Quotient / 2
 0E34 20C3          JR     NZ,0DF9H      ;Exp = 0 ? Nein: nächstes Bit
                                         ;verrechnen
 0E36 C3B207        JP     07B2H         ;Ja: OV-Error

: UPRO für DDIV
: Mantisse von X nach 414AH bis 4150H retten und X als Ergebnis auf 0 setzen

 0E39 79            LD     A,C           ;A = MSB (Y)
 0E3A 322D41        LD     (412DH),A     ;MSB (Y) zurückschreiben
 0E3D 2B            DEC    HL            ;HL : MSB (X)
 0E3E 115041        LD     DE,4150H      ;DE = Zeiger auf Zwischenspeicher
 0E41 010007        LD     BC,0700H      ;7 Bytes kopieren, Mantisse von X
                                         ;mit 00H füllen
 0E44 7E            LD     A,(HL)        ;A = Byte von X
 0E45 12            LD     (DE),A        ;Byte nach (DE) retten
 0E46 71            LD     (HL),C        ;und Mantisse löschen
 0E47 1B            DEC    DE            ;Zeiger -1
 0E48 2B            DEC    HL
 0E49 05            DEC    B             ;Zähler -1
 0E4A 20F8          JR     NZ,0E44H      ;Nächstes Byte
 0E4C C9            RET
```

```
; X = X * 10 (DBL)

  0E4D CDFC09         CALL    09FCH        ;Y = X
  0E50 EB             EX      DE,HL        ;HL+1 : Exp (X)
  0E51 2B             DEC     HL           ;HL : Exp (X)
  0E52 7E             LD      A,(HL)       ;A = Exp (X)
  0E53 B7             OR      A            ;X = 0 ?
  0E54 C8             RET     Z            ;Ja: Ergebnis = 0
  0E55 C602           ADD     A,02H        ;A = Exp (X) + 2
  0E57 DAB207         JP      C,07B2H      ;OV-Error bei Überlauf
  0E5A 77             LD      (HL),A       ;Exp zurück: X = X * 4
  0E5B E5             PUSH    HL           ;Zeiger retten
  0E5C CD770C         CALL    0C77H        ;X = X + Y
                                           ;(X * 4 + X ergibt X * 5)
  0E5F E1             POP     HL           ;Zeiger zurück
  0E60 34             INC     (HL)         ;Exp +1 entspricht X = X * 2
                                           ;(X * 5 * 2 ergibt X * 10)
  0E61 C0             RET     NZ           ;Fertig wenn kein Überlauf
  0E62 C3B207         JP      07B2H        ;sonst OV-Error

; Umwandlung eines Strings in eine Zahl (DBL)
; (wie VAL-Funktion)
; I: HL = Zeiger auf String
; O: X = Zahl (DBL)

  0E65 CD7807         CALL    0778H        ;X = 0
  0E68 CDEC0A         CALL    0AECH        ;VT auf DBL setzen
  0E6B F6AF           OR      0AFH         ;Flag <> 0

; Umwandlung eines Strings in eine Zahl passenden Typs (INT,SNG,DBL)
; (wie VAL-Funktion)
; I: HL = Zeiger auf String
; O: X = Zahl

 *0E6C   AF           XOR     A            ;Flag = 0
                                           ;Es wird erst probiert einen INT-
                                           ;Wert zu erzeugen. Bei Überlauf
                                           ;erfolgt eine automatische Um-
                                           ;wandlung ins SNG- bzw. DBL-
                                           ;Format
  0E6D EB             EX      DE,HL        ;DE = Zeiger auf String
  0E6E 01FF00         LD      BC,00FFH     ;B = 00H (Anzahl der Nachkomma-
                                           ;stellen)
                                           ;C = FFH (Kommaflag, siehe 0EE4H
                                           ;und 0F29H ff)
  0E71 60             LD      H,B          ;HL = 0000H
  0E72 68             LD      L,B          ;(HL ist Anfangswert)
  0E73 CC9A0A         CALL    Z,0A9AH      ;X (INT) auf 0 setzen
                                           ;wenn Flag = 0
```

```
 0E76 EB             EX     DE,HL       ;HL = Zeiger, DE = 0000H
 0E77 7E             LD     A,(HL)      ;A = 1. Zeichen
 0E78 FE2D           CP     2DH         ;Negatives Vorzeichen ? (Ja: Z=1)
 0E7A F5             PUSH   AF          ;Vorzeichen retten
 0E7B CA830E         JP     Z,0E83H     ;Sprung wenn Sign angegeben
 0E7E FE2B           CP     2BH         ;Positives Vorzeichen ?
 0E80 2801           JR     Z,0E83H     ;Ja: Sprung
 0E82 2B             DEC    HL          ;Nein: Zeiger -1 für RST 10H
 0E83 D7             RST    10H         ;A = Nächstes Zeichen
                                        ;Ziffer gefunden ?
 0E84 DA290F         JP     C,0F29H     ;Ja: weiter bei 0F29H
 0E87 FE2E           CP     2EH         ;'.' ?
 0E89 CAE40E         JP     Z,0EE4H     ;Ja: weiter bei 0EE4H
 0E8C FE45           CP     45H         ;'E' ?
 0E8E 2814           JR     Z,0EA4H     ;Ja: weiter bei 0EA4H mit Z = 1
 0E90 FE25           CP     25H         ;'%' (INT-Kennung) ?
 0E92 CAEE0E         JP     Z,0EEEH     ;Ja: weiter bei 0EEEH
 0E95 FE23           CP     23H         ;'#' (DBL-Kennung) ?
 0E97 CAF50E         JP     Z,0EF5H     ;Ja: weiter bei 0EF5H
 0E9A FE21           CP     21H         ;'!' (SNG-Kennung) ?
 0E9C CAF60E         JP     Z,0EF6H     ;Ja: weiter bei 0EF6H
 0E9F FE44           CP     44H         ;'D' ?
 0EA1 2024           JR     NZ,0EC7H    ;Nein: Weder Ziffer noch
                                        ;Sonderzeichen erkannt
                                        ;-> Ende des Zahlenstrings
                                        ;erreicht
 0EA3 B7             OR     A           ;Ja: Z = 0 setzen

; 'E' (Z=1) und 'D' (Z=0)

 0EA4 CDFB0E         CALL   0EFBH       ;X ins SNG- (Z=1) oder DBL- (Z=0)
                                        ;Format umwandeln
 0EA7 E5             PUSH   HL          ;Zeiger retten
 0EA8 21BD0E         LD     HL,0EBDH    ;RET-Adresse auf 0EBDH stellen
 0EAB E3             EX     (SP),HL     ;und Zeiger zurück nach HL
 0EAC D7             RST    10H         ;A = nächstes Zeichen nach 'E'
                                        ;bzw 'D'
 0EAD 15             DEC    D           ;D = FFH
 0EAE FECE           CP     0CEH        ;'-' (Basic-Token) ?
 0EB0 C8             RET    Z           ;Ja: weiter bei 0EBDH
 0EB1 FE2D           CP     2DH         ;'-' ?
 0EB3 C8             RET    Z           ;Ja: weiter bei 0EBDH
 0EB4 14             INC    D           ;D = 00H
 0EB5 FECD           CP     0CDH        ;'+' (Basic-Token) ?
 0EB7 C8             RET    Z           ;Ja: weiter bei 0EBDH
 0EB8 FE2B           CP     2BH         ;'+' ?
 0EBA C8             RET    Z           ;Ja: weiter bei 0EBDH
 0EBB 2B             DEC    HL          ;Zeiger -1 (Da bei RST 10H der
                                        ;Zeiger erhöht wurde)
 0EBC F1             POP    AF          ;RET-Adr. (0EBDH) vom Stack
                                        ;entfernen
```

```
 0EBD D7           RST    10H        :A = Exponentenzeichen
                                     :Ziffer gefunden ?
 0EBE DA940F       JP     C,0F94H    :Ja: weiter bei 0F94H
                                     :Nein: Exponent zuende
 0EC1 14           INC    D          :Exponent negativ ? (war D=FFH ?)
 0EC2 2003         JR     NZ,0EC7H   :Nein: weiter bei 0EC7H
 0EC4 AF           XOR    A          :Ja: Exponenten negieren
 0EC5 93           SUB    E          :A = 00H - E
 0EC6 5F           LD     E,A        :E = richtiger Exponent

; Zahl in X ist fertig: Exponent bzw. Kommastelle und Vorzeichen verarbeiten

 0EC7 E5           PUSH   HL         :Zeiger retten
 0EC8 7B           LD     A,E        :A = Exponent
 0EC9 90           SUB    B          :A = Differenz zwischen Exponent
                                     :und Anzahl der Nachkommastellen
                                     :Ist die Anzahl der Nachkomma-
                                     :stellen göβer als der Exponent ?
 0ECA F40A0F       CALL   P,0F0AH    :Ja: X mit 10 multiplizieren
                                     :und Differenz -1
 0ECD FC180F       CALL   M,0F18H    :Nein: X durch 10 dividieren
                                     :und Differenz +1
 0ED0 20F8         JR     NZ,0ECAH   :Weiterrechnen bis die Differenz
                                     :gleich 0 ist
 0ED2 E1           POP    HL         :Zeiger zurück
 0ED3 F1           POP    AF         :Vorzeichen zurück
 0ED4 E5           PUSH   HL         :Zeiger retten
 0ED5 CC7B09       CALL   Z,097BH    :X = -X, wenn das Vorzeichen '-'
                                     :war
 0ED8 E1           POP    HL         :Zeiger zurück
 0ED9 E7           RST    20H        :TSTTYP
 0EDA E8           RET    PE         :Fertig wenn X im DBL-Format
 0EDB E5           PUSH   HL         :Zeiger retten
 0EDC 219008       LD     HL,0890H   :RET-Adr auf POP HL (für Zeiger
 0EDF E5           PUSH   HL         :zurück) setzen
 0EE0 CDA30A       CALL   0AA3H      :Ist X = -32768 ?
                                     :Ja: X ins INT-Format umwandeln
 0EE3 C9           RET               :RET und Zeiger zurück

; '.' gefunden

 0EE4 E7           RST    20H        :TSTTYP
 0EE5 0C           INC    C          :C = 00H, wenn zum erstenmal ein
                                     :Komma gefunden wurde, sonst ist
                                     :C > 0
 0EE6 20DF         JR     NZ,0EC7H   :Zahl fertig, wenn C > 0
 0EE8 DCFB0E       CALL   C,0EFBH    :Ist X noch im INT-Format ?
                                     :Wenn ja, dann X in SNG-Format
                                     :umwandeln
 0EEB C3830E       JP     0E83H      :Nächstes Zeichen holen
```

```
; '%' (INT-Kennung) gefunden

  0EEE E7                  RST     20H         :TSTTYP
  0EEF F29719              JP      P,1997H     :SN-Error, wenn X bereits im SNG-
                                               :oder DBL-Format ist
  0EF2 23                  INC     HL          :Zeiger +1
  0EF3 18D2                JR      0EC7H       :Zahl ist fertig

; '#' (DBL-Kennung) gefunden

  0EF5 B7                  OR      A           :Z = 0

; '!' (SNG-Kennung) gefunden (Z = 1)

  0EF6 CDFB0E              CALL    0EFBH       :Zahl ins SNG- (Z = 1) oder
                                               :DBL- (Z = 0) Format umwandeln
  0EF9 18F7                JR      0EF2H       :Zeiger +1, Zahl ist fertig

; Zahlenumwandlung in SNG oder DBL
; I: Z = 1 : X = CSNG ( X )
;    Z = 0 : X = CDBL ( X )
; O: -

  0EFB E5                  PUSH    HL          :Register retten
  0EFC D5                  PUSH    DE
  0EFD C5                  PUSH    BC
  0EFE F5                  PUSH    AF
  0EFF CCB10A              CALL    Z,0AB1H     :CSNG, wenn Z = 1
  0F02 F1                  POP     AF          :Flags zurück
  0F03 C4DB0A              CALL    NZ,0ADBH    :CDBL, wenn Z = 0
  0F06 C1                  POP     BC          :Register zurück
  0F07 D1                  POP     DE
  0F08 E1                  POP     HL
  0F09 C9                  RET

; X = X * 10 (SNG,DBL)
; Typrichtige Multiplikation von X mit 10
; Wird bei der Verarbeitung des Exponenten bzw der Nachkommastellen verwendet
; I: X       = Zahl (SNG- oder DBL-Format)
; O: X       = X * 10
;    A       = A - 1 (für Exponenten- und Kommastellenverarbeitung)

  0F0A C8                  RET     Z           :Fertig wenn Z = 1 (Differenz
                                               :zwischen Exponent und
                                               :Nachkommastellen = 0)
  0F0B F5                  PUSH    AF          :Differenz retten
  0F0C E7                  RST     20H         :TSTTYP
  0F0D F5                  PUSH    AF          :Flags retten
  0F0E E43E09              CALL    PO,093EH    :X = X * 10 (SNG)
  0F11 F1                  POP     AF          :Flags zurück
  0F12 EC4D0E              CALL    PE,0E4DH    :X = X * 10 (DBL)
  0F15 F1                  POP     AF          :Differenz zurück
  0F16 3D                  DEC     A           :Differenz -1
  0F17 C9                  RET
```

```
: X = X / 10 (SNG.DBL)
: Typrichtige Division von X durch 10
: Gleiche Parameter wie bei X = X * 10, jedoch wird A um 1 erhöht

  0F18 D5              PUSH   DE           ;Register retten
  0F19 E5              PUSH   HL
  0F1A F5              PUSH   AF
  0F1B E7              RST    20H          ;TSTTYP
  0F1C F5              PUSH   AF           ;Flags retten
  0F1D E49708          CALL   PO,0897H     ;X = X / 10 (SNG)
  0F20 F1              POP    AF           ;Flags zurück
  0F21 ECDC0D          CALL   PE,0DDCH     ;X = X / 10 (DBL)
  0F24 F1              POP    AF           ;Register zurück
  0F25 E1              POP    HL
  0F26 D1              POP    DE
  0F27 3C              INC    A  -         ;Differenz +1
  0F28 C9              RET

: Ziffer verarbeiten (CY = 1 wegen RST 10H vorher)

  0F29 D5              PUSH   DE           ;Exponentenflags retten
  0F2A 78              LD     A,B          ;A = Anzahl der Nachkommastellen
  0F2B 89              ADC    A,C          ;C = FFH, wenn noch keine
                                           ;Nachkommastellen erkannt wurden:
                                           ;FFH + 1 (CY) ergibt 0, also wird
                                           ;zu A nichts hinzuaddiert.
                                           ;Im anderen Fall ist C = 00H
                                           ;+ 1 (CY) ergibt 1 -> die Anzahl
                                           ;der Nachkommastellen wird um 1
                                           ;erhöht
  0F2C 47              LD     B,A          ;B = Nachkommastellen
  0F2D C5              PUSH   BC           ;BC retten
  0F2E E5              PUSH   HL           ;Zeiger retten
  0F2F 7E              LD     A,(HL)       ;A = Ziffer (ASCII-Wert)
  0F30 D630            SUB    30H          ;A = Ziffer (Zahlenwert 0 bis 9)
  0F32 F5              PUSH   AF           ;Ziffernwert retten
  0F33 E7              RST    20H          ;TSTTYP, Ist X noch INT-Format ?
  0F34 F25D0F          JP     P,0F5DH      ;Nein: Weiter bei 0F5DH

: Neue Ziffer in INT-Zahl einarbeiten

  0F37 2A2141          LD     HL,(4121H)   ;HL = INT-Zahl
  0F3A 11CD0C          LD     DE,0CCDH     ;DE = 3277 (ca. 32767/10)
  0F3D DF              RST    18H          ;Ist die Zahl schon jetzt größer
                                           ;als 3277 ?
  0F3E 3019            JR     NC,0F59H     ;Ja: Durch die neue Stelle würde
                                           ;der Zahlenwert aus dem
                                           ;INT-Bereich herauskommen -> die
                                           ;Zahl muß ins SNG-Format
                                           ;umgewandelt werden
```

```
; Adresse der Variablen in (PTZ) ermitteln und Variable erzeugen,
; falls sie noch nicht existiert
; I: PTZ zeigt auf einen Variablennamen
; O: DE = Adresse der gesuchten Variablen
;       (= 0000H wenn die Variable nicht existiert)

*260D  AF        XOR    A            ;A = 0 für Adressen-Suche
 260E  32AE40    LD     (40AEH),A    ;Flag abspeichern
 2611  46        LD     B,(HL)       ;B = 1. Buchstabe des Namens
 2612  CD3D1E    CALL   1E3DH        ;Ist das Zeichen in (HL) ein
                                     ;Großbuchstabe ?
 2615  DA9719    JP     C,1997H      ;Nein: SN-Error
 2618  AF        XOR    A            ;A = 00
 2619  4F        LD     C,A          ;C = Default 2. Zeichen des
                                     ;Namens
 261A  D7        RST    10H          ;2. Zeichen angegeben ?
 261B  3805      JR     C,2622H      ;Sprung wenn das 2. Zeichen eine
                                     ;Ziffer ist
 261D  CD3D1E    CALL   1E3DH        ;Großbuchstabe angegeben ?
 2620  3809      JR     C,262BH      ;Nein: Variablennamen bei einem
                                     ;Zeichen belassen
 2622  4F        LD     C,A          ;C = 2. Zeichen
 2623  D7        RST    10H          ;Nächstes Zeichen holen
 2624  38FD      JR     C,2623H      ;Ziffer ? Ja: Zeichen übergehen
 2626  CD3D1E    CALL   1E3DH        ;Großbuchstabe ?
 2629  30F8      JR     NC,2623H     ;Ja: Zeichen übergehen
 262B  115226    LD     DE,2652H     ;RET-Adr auf 2652H setzen
 262E  D5        PUSH   DE
 262F  1602      LD     D,02H        ;D = 2 (Typcode für Integer)
 2631  FE25      CP     25H          ;'%' (INT-Kennung) gefunden ?
 2633  C8        RET    Z            ;Ja: D ist Typcode
 2634  14        INC    D            ;D + 1 (D = 3)
 2635  FE24      CP     24H          ;'$' (STR-Kennung) gefunden ?
 2637  C8        RET    Z            ;Ja: D ist Typcode
 2638  14        INC    D            ;D + 1 (D = 4)
 2639  FE21      CP     21H          ;'!' (SNG-Kennung) gefunden ?
 263B  C8        RET    Z            ;Ja: D ist Typcode
 263C  1608      LD     D,08H        ;D = 8 (Typcode für Double)
 263E  FE23      CP     23H          ;'#' (DBL-Kennung) gefunden ?
 2640  C8        RET    Z            ;Ja: D ist Typcode

; Kein Typcode angegeben
; Typcode aus der DEF-Tabelle holen

 2641  78        LD     A,B          ;A = 1. Buchstabe
 2642  D641      SUB    41H          ;A = Offset für Typcodetabelle
 2644  E67F      AND    7FH          ;Höchstes Bit ausbleden
 2646  5F        LD     E,A          ;DE = Offset
 2647  1600      LD     D,00H
 2649  E5        PUSH   HL           ;PTZ retten
 264A  210141    LD     HL,4101H     ;HL = Zeiger auf Tabelle
 264D  19        ADD    HL,DE        ;Offset addieren
 264E  56        LD     D,(HL)       ;D = Typcode (Siehe DEFXXX)
 264F  E1        POP    HL           ;PTZ zurück
 2650  2B        DEC    HL           ;PTZ -1
 2651  C9        RET                 ;RET nach 2652H
```

```
: AND / OR verarbeiten

  25E9 C5           PUSH   BC          ;Prioritätscode retten
  25EA CD7F0A       CALL   0A7FH       ;HL = X = CINT(X) = 2. Argument
  25ED F1           POP    AF          ;Prioritätscode zurück
  25EE D1           POP    DE          ;DE = 1. Argument
  25EF 01FA27       LD     BC,27FAH    ;RET-Adr auf 27FAH setzen
  25F2 C5           PUSH   BC          ;(X = HL (INT) )
  25F3 FE46         CP     46H         ;Prioritätscode = 46H (OR) ?
  25F5 2006         JR     NZ,25FDH    ;Nein: AND bei 25FDH ausführen

: OR

  25F7 7B           LD     A,E         ;Beide Argumente mit OR
  25F8 B5           OR     L           ;verknüpfen
  25F9 6F           LD     L,A
  25FA 7C           LD     A,H
  25FB B2           OR     D
  25FC C9           RET                ;Rücksprung nach 27FAH

: AND

  25FD 7B           LD     A,E         ;Beide Argumente mit AND
  25FE A5           AND    L           ;verknüpfen
  25FF 6F           LD     L,A
  2600 7C           LD     A,H
  2601 A2           AND    D
  2602 C9           RET                ;Rücksprung nach 27FAH

: Return nach DIM

  2603 2B           DEC    HL          ;PTZ -1
  2604 D7           RST    10H         ;Befehlsende erreicht ?
  2605 C8           RET    Z           ;Ja: Fertig
  2606 CF           RST    08H         ;Nein: nächste Variable muß durch
  2607 2C           DEFB   ','         ;Komma abgetrennt sein

: DIM

  2608 010326       LD     BC,2603H    ;RET-Adr auf 2603H setzen
  260B C5           PUSH   BC
  260C F6AF         OR     0AFH        ;A <> 0 für DIM
```

```
0F40 54              LD      D,H           ;DE = Zahl
0F41 5D              LD      E,L
0F42 29              ADD     HL,HL         ;HL = HL * 2  = Zahl *  2
0F43 29              ADD     HL,HL         ;HL = HL * 2  = Zahl *  4
0F44 19              ADD     HL,DE         ;HL = HL + DE = Zahl *  5
0F45 29              ADD     HL,HL         ;HL = HL * 2  = Zahl * 10
0F46 F1              POP     AF            ;Ziffer zurück
0F47 4F              LD      C,A           ;BC = Ziffernwert
                                           ;(B = Nachkommastellen = 0, da
                                           ;INT-Format)
0F48 09              ADD     HL,BC         ;HL = Zahl + Neue Ziffer
0F49 7C              LD      A,H           ;Ist die neue Zahl > 32767 ?
0F4A B7              OR      A
0F4B FA570F          JP      M,0F57H       ;Ja: Zahl ins SNG-Format
                                           ;umwandeln
0F4E 222141          LD      (4121H),HL    ;X = Neue Zahl
0F51 E1              POP     HL            ;Zeiger zurück
0F52 C1              POP     BC            ;Kommastelle zurück
0F53 D1              POP     DE            ;Exponentenflag zurück
0F54 C3830E          JP      0E83H         ;Nächste Ziffer holen

; Überlauf bei INT

0F57 79              LD      A,C           ;A = Ziffernwert
0F58 F5              PUSH    AF            ;Ziffer retten
0F59 CDCC0A          CALL    0ACCH         ;X = CSNG (X)
0F5C 37              SCF                   ;CY = 1 für SNG-Verarbeitung

; Neue Ziffer in SNG- (CY = 1) oder DBL-Zahl (CY = 0) einarbeiten

0F5D 3018            JR      NC,0F77H      ;Sprung wenn DBL
0F5F 017494          LD      BC,9474H      ;BCDE = 1E+6
0F62 110024          LD      DE,2400H
0F65 CD0C0A          CALL    0A0CH         ;X und BCDE vergleichen
                                           ;Ist X schon jetzt >= 1E+6 ?
0F68 F2740F          JP      P,0F74H       ;Ja: X in DBL umwandeln, da mit
                                           ;der neuen Ziffer X mehr als
                                           ;6 Stellen hätte und damit aus
                                           ;dem SNG-Bereich kommen würde
0F6B CD3E09          CALL    093EH         ;X = X * 10
0F6E F1              POP     AF            ;A = Ziffernwert
0F6F CD890F          CALL    0F89H         ;X = X + A (SNG)
0F72 18DD            JR      0F51H         ;Nächste Ziffer holen

; Überlauf bei SNG

0F74 CDE30A          CALL    0AE3H         ;X = CDBL (X)
```

```
; Neue Ziffer in DBL-Zahl einarbeiten

  0F77 CD4D0E          CALL    0E4DH          ;X = X * 10 (DBL)
  0F7A CDFC09          CALL    09FCH          ;Y = X
  0F7D F1              POP     AF             ;A = Ziffernwert
  0F7E CD6409          CALL    0964H          ;X = A
  0F81 CDE30A          CALL    0AE3H          ;X = CDBL (X)
  0F84 CD770C          CALL    0C77H          ;X = X + Y
  0F87 18C8            JR      0F51H          ;Nächste Ziffer holen

; X = X + A (SNG)

  0F89 CDA409          CALL    09A4H          ;(SP) = X
  0F8C CD6409          CALL    0964H          ;X = A

; X = X + (SP) (SNG)

  0F8F C1              POP     BC             ;BCDE = (SP)
  0F90 D1              POP     DE
  0F91 C31607          JP      0716H          ;X = X + BCDE

; Ziffer nach 'E' bzw 'D' gefunden

  0F94 7B              LD      A,E            ;A = bisheriger Exponent
  0F95 FE0A            CP      0AH            ;Ist der Exponent >= 10 ?
                                              ;Sind schon zwei Exponenten-
                                              ;stellen erkannt worden ?
                                              ;(Eine Stelle kann nur 9 ergeben)
  0F97 3009            JR      NC,0FA2H       ;Ja: Exponenten auf 48 stellen,
                                              ;und so einen Überlauf erzwingen
  0F99 07              RLCA                   ;A = A * 2 = Exponent *  2
  0F9A 07              RLCA                   ;A = A * 2 = Exponent *  4
  0F9B 83              ADD     A,E            ;A = A + E = Exponent *  5
  0F9C 07              RLCA                   ;A = A * 2 = Exponent * 10
  0F9D 86              ADD     A,(HL)         ;Neue Exponentenziffer verrechen
  0F9E D630            SUB     30H            ;30H abziehen, da der ASCII-Wert
                                              ;verrechnet wurde (Garantiert
                                              ;positives Ergebnis, da (HL) im
                                              ;Bereich von 30H bis 39H liegt)
  0FA0 5F              LD      E,A            ;E = Neuer Exponent
  0FA1 FA1E32          JP      M,321EH        ;-- (positives Ergebnis !)
 *0FA2   1E32          LD      E,32H          ;Exponent = 48 bei Überlauf
  0FA4 C3BD0E          JP      0EBDH          ;Nächste Exponentenziffer holen

; Print 'in' und Zahl in HL (Routine für Error und Break)

  0FA7 E5              PUSH    HL             ;Zahl retten
  0FA8 212419          LD      HL,1924H       ;(HL) = Text 'in '
  0FAB CDA728          CALL    28A7H          ;Text ausgeben
  0FAE E1              POP     HL             ;Zahl zurück
```

```
; Print HL (Routine für Zeilennummernausgabe bei LIST)

  0FAF CD9A0A          CALL    0A9AH        ;Zahl als INT nach X
  0FB2 AF              XOR     A            ;A = 00H -> keine Formatierung
  0FB3 CD3410          CALL    1034H        ;Formatierungsbyte abspeichern
                                            ;und Vorzeichen löschen
  0FB6 B6              OR      (HL)         ;A = 20H (A.7 = 0)
  0FB7 CDD90F          CALL    0FD9H        ;Unformatierten String erzeugen
  0FBA C3A628          JP      28A6H        ;und ausgeben

; Umwandlung von X in einen unformatierten String (für PRINT)
; (wie STR$)
; I: X = Zahl
; O: HL = Zeiger auf String (= 4130H)

  0FBD AF              XOR     A            ;Formatbyte löschen

; Umwandlung von X in einen formatierten String (für PRINT USING)
; (wie STR$)
; I: X = Zahl
;    A = Formatiercode: Bit 7 = 1: Formatierung ausführen
;                       Bit 6 = 1: ',' zur Tausenderstellentrennung ausgegeben
;                       Bit 5 = 1: Führende Leerstellen mit '*' auffüllen
;                       Bit 4 = 1: '$' vor der Zahl ausgegeben
;                       Bit 3 = 1: Vorzeichen (auch '+') mitausgeben
;                       Bit 2 = 1: Vorzeichen hinter der Zahl ausgeben
;                       Bit 1 = -: unbenutzt
;                       Bit 0 = 1: Zehnerexponenten mitausgeben
;
;    B = Anzahl der Vorkommastellen
;    C = Anzahl der Nachkommastellen + 1 (für Dezimalpunkt)
;
; O: HL = Zeiger auf Stringanfang (= 4130H)
;    DE = Zeiger auf Stringende

  0FBE CD3410          CALL    1034H        ;Formatbyte abspeichern
                                            ;Vorzeichenstelle im Buffer
                                            ;löschen und HL = 4130H (Buffer-
                                            ;anfang) setzen
  0FC1 E608            AND     08H          ;Ist das Vorzeichen gefordert ?
  0FC3 2802            JR      Z,0FC7H      ;Nein: weiter bei 0FC7H
  0FC5 362B            LD      (HL),2BH     ;Ja: erstmal '+' einsetzen
  0FC7 EB              EX      DE,HL        ;HL nach DE retten
  0FC8 CD9409          CALL    0994H        ;TEST1
  0FCB EB              EX      DE,HL        ;HL zurück
  0FCC F2D90F          JP      P,0FD9H      ;Vorzeichen belassen, wenn
                                            ;X positiv ist
  0FCF 362D            LD      (HL),2DH     ;sonst '-' als Vorzeichen
                                            ;einsetzen
  0FD1 C5              PUSH    BC           ;BC retten
  0FD2 E5              PUSH    HL           ;HL retten
```

```
 0FD3 CD7B09            CALL    097BH         ;X = -X
                                              ;(X wird als positiver Wert ver-
                                              ;arbeitet, da das Vorzeichen
                                              ;schon im Buffer steht)
 0FD6 E1                POP     HL            ;HL zurück
 0FD7 C1                POP     BC            ;BC zurück
 0FD8 B4                OR      H             ;Z = 0
                                              ;X ist jetzt positiv
                                              ;Z = 1 wenn X gleich 0 ist
 0FD9 23                INC     HL            ;Zeiger +1
 0FDA 3630              LD      (HL),30H      ;'0' in Buffer setzen
 0FDC 3AD840            LD      A,(40D8H)     ;A = Formatbyte
 0FDF 57                LD      D,A           ;Formatbyte nach D retten
 0FE0 17                RLA                   ;CY = Bit 7
 0FE1 3AAF40            LD      A,(40AFH)     ;A = VT
 0FE4 DA9A10            JP      C,109AH       ;Sprung wenn Formatierung
                                              ;gefordert
 0FE7 CA9210            JP      Z,1092H       ;Fertig wenn Zahl = 0
                                              ;(Z=1 von 0FC8H)
 0FEA FE04              CP      04H           ;Ist X im INT-Format (VT < 4) ?
 0FEC D23D10            JP      NC,103DH      ;Nein: weiter bei 103DH

; INT-Zahl in String umwandeln (ohne Formatierung)

 0FEF 010000            LD      BC,0000H      ;B = 0: Keinen Dezimalpunkt er-
                                              ;zeugen
                                              ;C = 0: Keine Tausendertrennung
 0FF2 CD2F13            CALL    132FH         ;Zahl in unformatierten String
                                              ;mit 5 Ziffern umwandeln
                                              ;(incl. führende Nullen)

; Führende Nullen löschen bzw durch '*' ersetzen

 0FF5 213041            LD      HL,4130H      ;HL = Bufferzeiger
 0FF8 46                LD      B,(HL)        ;B = Vorzeichen (' ' oder '-')
 0FF9 0E20              LD      C,20H         ;C = ' '
 0FFB 3AD840            LD      A,(40D8H)     ;A = Formatbyte
 0FFE 5F                LD      E,A           ;E = Formatbyte
 0FFF E620              AND     20H           ;Leerzeichen mit '*' füllen ?
 1001 2807              JR      Z,100AH       ;Nein: weiter bei 100AH
 1003 78                LD      A,B           ;Ja: A = Vorzeichen
 1004 B9                CP      C             ;Ist das Vorzeichen = ' ' ?
 1005 0E2A              LD      C,2AH         ;C = '*'
 1007 2001              JR      NZ,100AH      ;Nein: '-' vorhanden
 1009 41                LD      B,C           ;Ja: Vorzeichen (' ') durch '*'
                                              ;ersetzen
 100A 71                LD      (HL),C        ;' ' oder '*' in den Buffer
                                              ;schreiben
```

```
100B D7          RST   10H          ;A = nächstes Zeichen
100C 2814        JR    Z,1022H      ;Sprung wenn String zuende
100E FE45        CP    45H          ;'E' gefunden ?
1010 2810        JR    Z,1022H      ;Ja: Ende gefunden ('E' gehört
                                    ;nicht zum INT-Format)
1012 FE44        CP    44H          ;'D' gefunden ?
1014 280C        JR    Z,1022H      ;Ja: Ende gefunden
1016 FE30        CP    30H          ;Führende 0 gefunden ?
1018 28F0        JR    Z,100AH      ;Ja: Durch ' ' oder '*' ersetzen
101A FE2C        CP    2CH          ;',' gefunden ?
101C 28EC        JR    Z,100AH      ;Ja: Durch ' ' oder '*' ersetzen
101E FE2E        CP    2EH          ;'.' gefunden ?
1020 2003        JR    NZ,1025H     ;Nein: weiter bei 1025H
1022 2B          DEC   HL           ;Ja: Dezimalpunkt
1023 3630        LD    (HL),30H     ;durch '0' ersetzen
1025 7B          LD    A,E -        ;A = Formatbyte
1026 E610        AND   10H          ;'$' vor Zahl ?
1028 2803        JR    Z,102DH      ;Nein: weiter bei 102DH
102A 2B          DEC   HL           ;Ja: '$' einsetzen
102B 3624        LD    (HL),24H
102D 7B          LD    A,E          ;A = Formatbyte
102E E604        AND   04H          ;Vorzeichen hinter der Zahl
                                    ;ausgeben ?
1030 C0          RET   NZ           ;Ja: String ist fertig
1031 2B          DEC   HL           ;Nein: Vorzeichen wieder vor der
1032 70          LD    (HL),B       ;Zahl einsetzen
1033 C9          RET

; Formatbyte retten, HL auf Bufferanfang setzen und Vorzeichen löschen

1034 32D840      LD    (40D8H),A    ;Formatbyte retten
1037 213041      LD    HL,4130H     ;HL : Bufferanfang
103A 3620        LD    (HL),20H     ;Vorzeichen im Buffer löschen
103C C9          RET

; X ist im Fließkommaformat (A = VT)
; String ohne Formatierung erzeugen

103D FE05        CP    05H          ;CY = 1 wenn SNG, sonst CY = 0
103F E5          PUSH  HL           ;Zeiger retten
1040 DE00        SBC   A,00H        ;A = 3 wenn SNG, A = 8 wenn DBL
1042 17          RLA                ;* 2 ergibt die Anzahl der
                                    ;maximal zu erzeugenden Dezimal-
                                    ;stellen - 1
1043 57          LD    D,A          ;D = A
1044 14          INC   D            ;D = maximale Stellenzahl
                                    ;(7 für SNG und 17 für DBL)
1045 CD0112      CALL  1201H        ;X auf 6 bzw. 16 Stellen ska-
                                    ;lieren
                                    ;A = Exponentenoffset (= Anzahl
                                    ;der Kommaverschiebungen nach
                                    ;links während der Skalierung)
```

```
1048 010003          LD      BC,0300H     ;B = 3 (=Dezimalpunktposition+1)
                                          ;C = 0 (Keine Tausendertrennung)
104B 82              ADD     A,D          ;A = Exponentenoffset + maximale
                                          ;Stellenzahl = 10-Exponent + 2
104C FA5710          JP      M,1057H      ;weiter bei 1057H wenn der 10-
                                          ;Exponent < -2 ist
104F 14              INC     D            ;Ist die Stellenzahl + 1 kleiner
1050 BA              CP      D            ;als der 10-Exponent + 2 ?
1051 3004            JR      NC,1057H     ;Ja: weiter bei 1057H
1053 3C              INC     A            ;Nein: 10-Exponent + 3 = Dezimal-
                                          ;punktposition + 1
1054 47              LD      B,A          ;B = Dezimalpunktposition + 1
1055 3E02            LD      A,02H        ;A = 2 wegen SUB 02H -> Es wird
                                          ;kein 10-Exponent ausgegeben

1057 D602            SUB     02H          ;A = 10-Exponent
1059 E1              POP     HL           ;Bufferzeiger zurück
105A F5              PUSH    AF           ;10-Exponent retten
105B CD9112          CALL    1291H        ;',' und '.' setzen, B -1
105E 3630            LD      (HL),30H     ;'0' einsetzen
1060 CCC909          CALL    Z,09C9H      ;Zeiger +1 wenn Dezimalpunkt
                                          ;gesetzt wurde
1063 CDA412          CALL    12A4H        ;X in unformatierten String mit
                                          ;7 bzw. 17 Stellen (inkl. führen-
                                          ;der Nullen) umwandeln. Dezimal-
                                          ;punkt nach B Ziffern einsetzen
1066 2B              DEC     HL           ;Bufferzeiger -1
1067 7E              LD      A,(HL)       ;A = Zeichen
1068 FE30            CP      30H          ;nachfolgende Nullen ?
106A 28FA            JR      Z,1066H      ;Ja: Zeiger = Letztes Zeichen
                                          ;das nicht '0' ist (Nachfolgende
                                          ;Nullen werden durch den 10-Expo-
                                          ;nenten dargestellt)
106C FE2E            CP      2EH          ;Ist das letzte Zeichen der
                                          ;Dezimalpunkt ?
                                          ;Ja: Zeiger belassen -> Der De-
                                          ;zimalpunkt wird gelöscht
106E C4C909          CALL    NZ,09C9H     ;Nein: Zeiger +1
1071 F1              POP     AF           ;10-Exp zurück
1072 281F            JR      Z,1093H      ;Zahl fertig wenn der 10-Exp
                                          ;gleich 0 ist

: 10-Exponenten ausgeben
: A = 10-Exponent

1074 F5              PUSH    AF           ;10-Exp retten
1075 E7              RST     20H          ;TSTTYP (CY = 1 wenn SNG-Format)
1076 3E22            LD      A,22H        ;A = ASCII-Wert von 'D' / 2
1078 8F              ADC     A,A          ;A = A * 2 + CY
                                          ;A = 'E' wenn X im SNG-Format
                                          ;sonst A = 'D'
1079 77              LD      (HL),A       ;Korrekte 10-Exponentenkennung
                                          ;einsetzen
```

```
107A 23              INC    HL          :Zeiger +1
107B F1              POP    AF          :10-Exp zurück
107C 362B            LD     (HL),2BH    :Positives Vorzeichen annehmen
107E F28510          JP     P,1085H     :Vorzeichen ok wenn 10-Exp > 0
1081 362D            LD     (HL),2DH    :Sonst '-' einsetzen
1083 2F              CPL                :und den 10-Exp negieren
1084 3C              INC    A           :(also als positive Zahl
                                        :behandeln)
1085 062F            LD     B,2FH       :B = ASCII-Wert von '0' - 1
1087 04              INC    B           :B +1 (nächste Ziffer der Zehner-
                                        :stelle)
1088 D60A            SUB    0AH         :10 vom 10-Exp abziehen
108A 30FB            JR     NC,1087H    :Ziffer weiter erhöhen solange
                                        :der 10-Exp noch > 10 ist
108C C63A            ADD    A,3AH       :+3AH ergibt korrekten ASCII-Wert
                                        :der Einerstelle
108E 23              INC    HL          :Zeiger +1
108F 70              LD     (HL),B      :Zehnerstelle des Exp einsetzen
1090 23              INC    HL          :Zeiger +1
1091 77              LD     (HL),A      :Einerstelle einsetzen

: String durch Einsetzen von 00H abschließen

1092 23              INC    HL          :Zeiger +1
1093 3600            LD     (HL),00H    :mit 00H abschließen
1095 EB              EX     DE,HL       :DE = Endzeiger
1096 213041          LD     HL,4130H    :HL = Anfangszeiger
1099 C9              RET

: Formatierung gefordert
: A  = VT
: BC = Vor- und Nachkommastellenzähler
: D  = Formatbyte
: HL = Bufferzeiger

109A 23              INC    HL          :Zeiger +1
109B C5              PUSH   BC          :Vor- und Nachkommastellen-
                                        :zähler retten
109C FE04            CP     04H         :Ist X im INT-Format ?
109E 7A              LD     A,D         :A = Formatbyte
109F D20911          JP     NC,1109H    :Nein: weiter bei 1109H

: INT-Zahl in formatierten String umwandeln

10A2 1F              RRA                :Bit 0 des Formatbytes nach CY
10A3 DAA311          JP     C,11A3H     :weiter bei 11A3H wenn der 10-
                                        :Exponent ausgegeben werden soll
                                        :(Dazu muß X ins SNG-Format umge-
                                        :wandelt werden)
10A6 010306          LD     BC,0603H    :B = maximale Anzahl der Vor-
                                        :kommastellen + 1
                                        :C = Zähler für Tausender-
                                        :trennung
```

```
10A9 CD8912         CALL    1289H       ;C = 0 setzen wenn keine Tau-
                                        ;sendertrennung erwünscht ist
10AC D1             POP     DE          ;D = Anzahl der Vorkommastellen
                                        ;E = Anzahl der Nachkommastellen
10AD 7A             LD      A,D         ;A = Anzahl der Vorkommastellen
10AE D605           SUB     05H         ;Mehr als 4 Vorkommastellen ?
10B0 F46912         CALL    P,1269H     ;Ja: Entsprechende Anzahl füh-
                                        ;render Nullen einsetzen
10B3 CD2F13         CALL    132FH       ;X in String mit 5 Ziffern um-
                                        ;wandeln
10B6 7B             LD      A,E         ;A = Anzahl der Nachkommastellen
10B7 B7             OR      A           ;Keine Nachkommastellen ?
10B8 CC2F09         CALL    Z,092FH     ;Ja: Bufferzeiger -1
10BB 3D             DEC     A           ;Nachkommastellen erwünscht ?
                                        ;(A ist FFH wenn nicht !)
10BC F46912         CALL    P,1269H     ;Ja: Entsprechende Anzahl Nullen
                                        ;einsetzen
10BF E5             PUSH    HL          ;Bufferzeiger auf Ende des
                                        ;Strings retten
10C0 CDF50F         CALL    0FF5H       ;Führende Nullen durch löschen
                                        ;oder durch '*' ersetzen
10C3 E1             POP     HL          ;Bufferzeiger zurück
10C4 2802           JR      Z,10C8H     ;Sprung wenn das Vorzeichen vor
                                        ;der Zahl eingesetzt wurde
10C6 70             LD      (HL),B      ;Sonst jetzt das Vorzeichen
10C7 23             INC     HL          ;hinter der Zahl einsetzen
10C8 3600           LD      (HL),00H    ;String abschließen
10CA 212F41         LD      HL,412FH    ;HL = Bufferadresse - 1
10CD 23             INC     HL          ;Bufferzeiger +1
10CE 3AF340         LD      A,(40F3H)   ;A = LSB der Bufferadresse des
                                        ;Dezimalpunkts
10D1 95             SUB     L           ;- LSB des jetzigen Bufferzeigers
                                        ;= Anzahl der Vorkommastellen ab
                                        ;Bufferzeiger
10D2 92             SUB     D           ;Ist das gleich der Anzahl der
                                        ;gewünschten Vorkommastellen ?
10D3 C8             RET     Z           ;Ja: Fertig

; String im Buffer verschieben
; Eine Vorkommastelle löschen (d. h. Leerzeichen löschen)

10D4 7E             LD      A,(HL)      ;A = nächstes Bufferzeichen
10D5 FE20           CP      20H         ;führendes Leerzeichen ?
10D7 28F4           JR      Z,10CDH     ;Ja: Übergehen, nächstes Zeichen
10D9 FE2A           CP      2AH         ;'*' vor der Zahl ?
10DB 28F0           JR      Z,10CDH     ;Ja: Übergehen, nächstes Zeichen
10DD 2B             DEC     HL          ;Nein: Bufferzeiger -1
10DE E5             PUSH    HL          ;Bufferzeiger retten (Zeigt jetzt
                                        ;auf das Vorzeichen bzw. die
                                        ;erste Ziffer oder '$')
```

```
10DF F5          PUSH  AF          ;Zeichen im Stack ablegen
10E0 01DF10      LD    BC,10DFH    ;10DFH als RET-Adr setzen
10E3 C5          PUSH  BC          ;Und Zahlenstringanfang suchen
10E4 D7          RST   10H         ;Nächstes Zeichen holen
10E5 FE2D        CP    2DH         ;'-' gefunden ?
10E7 C8          RET   Z           ;Ja: Zeichen retten, nächstes
                                   ;Zeichen
10E8 FE2B        CP    2BH         ;'+' gefunden ?
10EA C8          RET   Z           ;Ja: Zeichen retten, nächstes
                                   ;Zeichen
10EB FE24        CP    24H         ;'$' gefunden ?
10ED C8          RET   Z           ;Ja: Zeichen retten, nächstes
                                   ;Zeichen
10EE C1          POP   BC          ;RET-Adr wieder löschen
10EF FE30        CP    30H         ;Führende Null gefunden ?
10F1 200F        JR    NZ,1102H    ;Nein: Feldüberlauf
10F3 23          INC   HL          ;Ja: Zeiger +1, Führende Null
                                   ;übergehen
10F4 D7          RST   10H         ;nächstes Zeichen holen
10F5 300B        JR    NC,1102H    ;Ist es eine Ziffer ?
                                   ;Nein: Feldüberlauf
10F7 2B          DEC   HL          ;Ja: Bufferzeiger -1 (also String
                                   ;eine Stelle 'früher' im Buffer
                                   ;beginnen lassen)
10F8 012B77      LD    BC,772BH    ;--
*10F9  2B        DEC   HL          ;Bufferzeiger -1
*10FA     77     LD    (HL),A      ;Zeichen wieder einsetzen
10FB F1          POP   AF          ;Zeichen aus Stack holen
10FC 28FB        JR    Z,10F9H     ;Stringanfang erreicht ?
                                   ;Nein: nächstes Zeichen einsetzen
10FE C1          POP   BC          ;Ja: Bufferzeiger aus dem Stack
                                   ;löschen
10FF C3CE10      JP    10CEH       ;Stellenzahl jetzt ok ?

: Feldüberlauf
: Es wurden mehr Vorkommastellen (bzw. Vorkommaziffern) erzeugt als gefordert

1102 F1          POP   AF          ;Zeichen aus Stack holen
1103 28FD        JR    Z,1102H     ;letztes Zeichen ?
                                   ;Nein: nächstes Zeichen holen
1105 E1          POP   HL          ;Bufferzeiger zurück (zeigt auf
                                   ;Stringanfang)
1106 3625        LD    (HL),25H    ;'%' als Überlaufkennung
                                   ;einsetzen
1108 C9          RET               ;Fertig


: Formatierung gefordert
: X ist im Fließkommaformat

1109 E5          PUSH  HL          ;Zeiger retten
110A 1F          RRA               ;Bit 0 nach CY schieben
110B DAAA11      JP    C,11AAH     ;Weiter bei 11AAH wenn
                                   ;Exponentenausgabe erwünscht
110E 2814        JR    Z,1124H     ;Weiter bei 1124H wenn X im
                                   ;SNG-Format ist
```

```
; X ist im DBL-Format
; Zahlenstring erzeugen ohne Exponentendarstellung
; (Dazu darf aber X nicht mehr als 16 Vorkommastellen haben
; d. h. X muß kleiner als 1D+16 sein)

  1110 118413          LD      DE,1384H     ;DE : 1D+16
  1113 CD490A          CALL    0A49H        ;CP X,(DE) = CP X, 1D+16
  1116 1610            LD      D,10H        ;D = Maximale Anzahl der auszu-
                                            ;gebenden Stellen (16)
  1118 FA3211          JP      M,1132H      ;weiter bei 1132H wenn X < 1D+16
                                            ;ist

; Feldüberlauf bei Fließkommazahl

  111B E1              POP     HL           ;Bufferzeiger zurück
  111C C1              POP     BC           ;Vor- und Nachkommastellen-
                                            ;zähler zurück
  111D CDBD0F          CALL    0FBDH        ;Unformatierten String erzeugen
  1120 2B              DEC     HL           ;Bufferzeiger -1
  1121 3625            LD      (HL),25H     ;'%' zur Überlaufkennung vor dem
                                            ;String einsetzen
  1123 C9              RET                  ;Fertig

; X ist im SNG-Format

  1124 010EB6          LD      BC,0B60EH    ;BCDE = 1E+16
  1127 11CA1B          LD      DE,1BCAH
  112A CD0C0A          CALL    0A0CH        ;CP X,BCDE = CP X, 1E+16
  112D F21B11          JP      P,111BH      ;Feldüberlauf wenn X > 1E+16
  1130 1606            LD      D,06H        ;D = Maximale Anzahl der auszu-
                                            ;gebenden Stellen (6)

  1132 CD5509          CALL    0955H        ;TEST2, Zahl = 0 ?
  1135 C40112          CALL    NZ,1201H     ;Nein: Zahl auf 6 bzw. 16 Stellen
                                            ;skalieren
  1138 E1              POP     HL           ;Bufferzeiger zurück
  1139 C1              POP     BC           ;Vor- und Nachkommastellenzahl
                                            ;zurück
  113A FA5711          JP      M,1157H      ;weiter bei der Skalierung er-
                                            ;weitert wurde

; Es wurde bei der Skalierung gekürzt (Keine Nachkommastellen)
; A = Exponentenoffset ( > 0 )

  113D C5              PUSH    BC           ;Stellenzahl retten
  113E 5F              LD      E,A          ;E = Exponentenoffset
  113F 78              LD      A,B          ;A = Anzahl der Vorkommastellen
  1140 92              SUB     D            ;- maximale Anzahl der auszu-
                                            ;gebenden Stellen
  1141 93              SUB     E            ;- Exponentenoffset
  1142 F46912          CALL    P,1269H      ;Entsprechende Anzahl führender
                                            ;Nullen einsetzen
  1145 CD7D12          CALL    127DH        ;Dezimalpunktposition und Zähler
                                            ;für die Tausendertrennung er-
                                            ;mitteln
```

```
1148 CDA412          CALL    12A4H        :Fließkommazahl in unformatierten
                                          :String umwandeln
114B B3              OR      E            :A = Exponentenoffset
114C C47712          CALL    NZ,1277H     :Entsprechende Anzahl nach-
                                          :folgender Nullen einsetzen
                                          :(Da der 10-Exponent nicht
                                          :ausgegeben wird)
114F B3              OR      E            :A = Exponentenoffset
1150 C49112          CALL    NZ,1291H     :Evtl. fehlenden Dezimalpunkt
                                          :einsetzen
1153 D1              POP     DE           :Stellenzahl zurück nach DE
1154 C3B610          JP      10B6H        :Stellenzahl ok ?

: Es wurde bei der Skalierung erweitert (Nachkommastellen vorhanden)
: A = Exponentenoffset ( < 0 )

1157 5F              LD      E,A          :E = Exponentenoffset
1158 79              LD      A,C          :A = Anzahl der gewünschten Nach-
                                          :kommastellen + 1
1159 B7              OR      A            :Nachkommastellen erwünscht ?
115A C4160F          CALL    NZ,0F16H     :Ja: A -1 (wegen Dezimalpunkt)
115D 83              ADD     A,E          :A = Nachkommastellen
                                          :+ Exponentenoffset
                                          := negative Anzahl der bei der
                                          :Skalierung zuviel erzeugten
                                          :Stellen
115E FA6211          JP      M,1162H      :A belassen wenn zuviel Stellen
                                          :erzeugt wurden
1161 AF              XOR     A            :sonst A = 0 setzen
1162 C5              PUSH    BC           :Stellenzahl retten
1163 F5              PUSH    AF           :Anzahl der zuviel erzeugten
                                          :Stellen retten
1164 FC180F          CALL    M,0F18H      :X = X/10, A +1:
1167 FA6411          JP      M,1164H      :Skalierung soweit zurücknehmen,
                                          :daß die gewünschte Anzahl von
                                          :Nachkommastellen wieder erreicht
                                          :wird
116A C1              POP     BC           :B = negative Anzahl der zuviel
                                          :erzeugten Nachkommastellen
116B 7B              LD      A,E          :A = Exponentenoffset
116C 90              SUB     B            :+ Anzahl der zuviel erzeugten
                                          :Nachkommastellen
                                          := jetziger Exponentenoffset
                                          :(nach Zurücknahme der Ska-
                                          :lierung)
116D C1              POP     BC           :Stellenzahl zurück
```

```
116E 5F             LD      E,A          ;E = Exponentenoffset ( < 0 !)
116F 82             ADD     A,D          ;Ist der Exponentenoffset + die
                                         ;Anzahl der maximal zu erzeugen-
                                         ;den Stellen kleiner als Null ?
                                         ;(d. h. Sind keine Vorkomma-
                                         ;stellen vorhanden ?)
1170 78             LD      A,B          ;A = Anzahl der auszugebenden
                                         ;Vorkommastellen
1171 FA7F11         JP      M,117FH      ;Ja: weiter bei 117FH
1174 92             SUB     D            ;Nein: A = Anzahl der auszugeben-
                                         ;den Vorkommastellen - Anzahl der
                                         ;maximal auszugebenden Stellen
1175 93             SUB     E            ;- Exponentenoffset
1176 F46912         CALL    P,1269H      ;Entsprechenden Anzahl führender
                                         ;Nullen in den Buffer setzen
1179 C5             PUSH    BC -         ;Stellezahl retten
117A CD7D12         CALL    127DH        ;Dezimalpunktposition und Zähler
                                         ;für die Tausendertrennung
                                         ;ermitteln
117D 1811           JR      1190H        ;weiter bei 1190H

; Es sind keine Vorkommastellen vorhanden (siehe 116EH ff)

117F CD6912         CALL    1269H        ;Gewünschte Vorkommafeldlänge
                                         ;durch Einsetzen führender Nullen
                                         ;simulieren
1182 79             LD      A,C          ;A = Anzahl der gewünschten Nach-
                                         ;kommastellen + 1
1183 CD9412         CALL    1294H        ;Dezimalpunkt setzen, C = B
1186 4F             LD      C,A          ;Nachkommafeldlänge nach C
                                         ;zurückschreiben
1187 AF             XOR     A            ;A = 0
1188 92             SUB     D            ;A = 0 - Anzahl der maximal zu
                                         ;erzeugenden Stellen
1189 93             SUB     E            ;+ Exponentenoffset (+ da E < 0)
118A CD6912         CALL    1269H        ;Entsprechende Anzahl Nullen nach
                                         ;dem Dezimalpunkt einsetzen
118D C5             PUSH    BC           ;Stellenzahl retten
118E 47             LD      B,A          ;B = 0 (A ist 0 wegen 1269H)
118F 4F             LD      C,A          ;C = 0
                                         ;-> Keinen Dezimalpunkt und keine
                                         ;Tausendertrennung einsetzen
1190 CDA412         CALL    12A4H        ;unformatierten String erzeugen
1193 C1             POP     BC           ;Stellenzahl zurück
1194 B1             OR      C            ;A = Anzahl der gewünschten Nach-
                                         ;kommastellen +1
                                         ;Nachkommastellen erwünscht ?
1195 2003           JR      NZ,119AH     ;Ja: Bufferzeiger belassen
                                         ;(Der Bufferzeiger zeigt auf das
                                         ;letzte Zeichen des Strings !)
1197 2AF340         LD      HL,(40F3H)   ;Nein: Bufferzeiger auf Dezimal-
                                         ;punktposition zeigen lassen
                                         ;(= Ende des Strings !)
```

```
119A 83             ADD     A,E          :A = Anzahl der gewünschten Nach-
                                         :kommastellen + 1
                                         :+ Exponentenoffset
119B 3D             DEC     A            :A -1 (wegen Nachkommastellen +1)
119C F46912         CALL    P,1269H      :Entsprechende Anzahl nachfol-
                                         :gender Nullen einsetzen
119F 50             LD      D,B          :D = Anzahl der gewünschten Vor-
                                         :kommastellen
11A0 C3BF10         JP      10BFH        :weiter bei 10BFH


: INT-Zahl mit Exponentenausgabe
: Dazu muß X ins SNG-Format gebracht werden

11A3 E5             PUSH    HL           :Bufferzeiger retten
11A4 D5             PUSH    DE           :Formatbyte retten
11A5 CDCC0A         CALL    0ACCH        :X = CSNG (X)
11A8 D1             POP     DE           :Formatbyte zurück
11A9 AF             XOR     A            :A = 0 (Z=1 da X im INT-Format)

: SNG- oder DBL-Zahl mit Exponentenausgabe

11AA CAB011         JP      Z,11B0H      :weiter bei 11B0H bei SNG-Zahl
11AD 1E10           LD      E,10H        :E = maximale Anzahl der auszu-
                                         :gebenden Stellen (16)
11AF 011E06         LD      BC,061EH     :--
*11B0  1E06         LD      E,06H        :E = maximale Anzahl der auszu-
                                         :gebenden Stellen (6)
11B2 CD5509         CALL    0955H        :TEST2, X = 0 ?
11B5 37             SCF                  :CY=1
11B6 C40112         CALL    NZ,1201H     :Nein: X auf 6 bzw. 16 Stellen
                                         :skalieren, CY = 0
11B9 E1             POP     HL           :Bufferzeiger zurück
11BA C1             POP     BC           :Stellenzahl zurück
11BB F5             PUSH    AF           :Exponentenoffset retten
11BC 79             LD      A,C          :A = Anzahl der Nachkommastellen
11BD B7             OR      A            :Nachkommastellen erwünscht ?
11BE F5             PUSH    AF           :Anzahl der Nachkommastellen
                                         :retten
11BF C4160F         CALL    NZ,0F16H     :Ja: A -1 (wegen Dezimalpunkt)
11C2 80             ADD     A,B          :+ Anzahl der Vorkommastellen
11C3 4F             LD      C,A          :C = Gesamtlänge
11C4 7A             LD      A,D          :A = Formatbyte
11C5 E604           AND     04H          :Vorzeichen hinter der Zahl aus-
                                         :geben ?
11C7 FE01           CP      01H          :Nein: CY = 1
11C9 9F             SBC     A,A          :A = FF wenn nicht
11CA 57             LD      D,A          :D = A
11CB 81             ADD     A,C          :A = Gesamtlänge belassen wenn
                                         :das Vorzeichen vor der Zahl aus-
                                         :gegeben werden soll
                                         :Sonst die Gesamtlänge um eins
                                         :erniedrigen
11CC 4F             LD      C,A          :C = Gesamtlänge
```

```
11CD 93         SUB     E           ;A = Gewünschte Gesamtlänge des
                                    ;Strings - maximale Anzahl der
                                    ;auszugebenden Stellen
                                    ;Ist die gewünschte Stellenzahl
                                    ;kleiner als die erzeugte
                                    ;Stellenzahl ?
11CE F5         PUSH    AF          ;Differenz retten
11CF C5         PUSH    BC          ;Gesamtlänge retten
11D0 FC180F     CALL    M,0F18H     ;Ja: X = X/10, A -1:
11D3 FAD011     JP      M,11D0H     ;X um die Differenz der Stellen-
                                    ;skalieren
11D6 C1         POP     BC          ;Gesamtlänge zurück
11D7 F1         POP     AF          ;Stellendifferenz zurück
11D8 C5         PUSH    BC          ;Gesamtlänge retten
11D9 F5         PUSH    AF          ;Stellendifferenz retten
11DA FADE11     JP      M,11DEH     ;weiter bei 11DEH wenn die
                                    ;Gesamtlänge > Stellenzahl war
11DD AF         XOR     A           ;sonst A = 0
11DE 2F         CPL                 ;A = positive Stellendifferenz
11DF 3C         INC     A
11E0 80         ADD     A,B         ;+ Anzahl der gewünschten Vor-
                                    ;kommastellen
11E1 3C         INC     A           ;+1
11E2 82         ADD     A,D         ;-1 (falls das Vorzeichen vor der
                                    ;Zahl ausgegeben wird)
11E3 47         LD      B,A         ;= Dezimalpunktposition
11E4 0E00       LD      C,00H       ;Keine Tausendertrennung erzeugen
11E6 CDA412     CALL    12A4H       ;Unformatierten String erzeugen
11E9 F1         POP     AF          ;Stellendifferenz zurück
11EA F47112     CALL    P,1271H     ;Entsprechenden Anzahl nachfol-
                                    ;gender Nullen einsetzen
11ED C1         POP     BC          ;Stellenzahlen zurück
11EE F1         POP     AF          ;A = Anzahl der Nachkommastellen
                                    ;Nachkommastellen erwünscht ?
11EF CC2F09     CALL    Z,092FH     ;Nein: Bufferzeiger -1
                                    ;(Dezimalpunkt wieder entfernen)
11F2 F1         POP     AF          ;Exponentenoffset zurück
                                    ;Ist X = 0 ? (siehe 11B5H ff)
11F3 3803       JR      C,11F8H     ;Ja: 10-Exp ist auch Null
11F5 83         ADD     A,E         ;Nein: maximale Stellenzahl
                                    ;addieren
11F6 90         SUB     B           ;Anzahl der bereits erzeugten
                                    ;Vorkommastellen abziehen
11F7 92         SUB     D           ;und die Addition von (11E2H)
                                    ;zurücknehmen
                                    ;= 10-Exponent

11F8 C5         PUSH    BC          ;Anzahl der Vorkommastellen
                                    ;retten
11F9 CD7410     CALL    1074H       ;10-Exp einsetzen
11FC EB         EX      DE,HL       ;HL = Endzeiger
11FD D1         POP     DE          ;D = Anzahl der gewünschten
                                    ;Vorkommastellen
11FE C3BF10     JP      10BFH       ;weiter bei 10BFH
```

```
: Skalierung:
: X auf 6 bzw. 16 Vorkommastellen bringen
: O: A = 10-Exponentenoffset

  1201 D5               PUSH    DE              ;DE retten
  1202 AF               XOR     A               ;Exponentenoffset = 00H
  1203 F5               PUSH    AF              ;Exponentenoffset retten
  1204 E7               RST     20H             ;TSTTYP. Ist X im SNG-Format ?
  1205 E22212           JP      PO,1222H        ;Ja: weiter bei 1222H

: X ist im DBL-Format

  1208 3A2441           LD      A,(4124H)       ;A = Exp (X)
  120B FE91             CP      91H             ;Ist X >= 2 hoch 16 ?
  120D D22212           JP      NC,1222H        ;Ja: weiter bei 1222H
  1210 116413           LD      DE,1364H        ;Nein: DE : 1D+10
  1213 212741           LD      HL,4127H        ;HL = Zeiger auf Y
  1216 CDD309           CALL    09D3H           ;(HL) = (DE): Y = 1D+10
  1219 CDA10D           CALL    0DA1H           ;X = X * Y = X * 1D+10
  121C F1               POP     AF              ;Exponentenoffset zurück
  121D D60A             SUB     0AH             ;Expoffset -10 (10 Dezimalstellen
                                                ;verschoben)
  121F F5               PUSH    AF              ;Exponentenoffset retten
  1220 18E6             JR      1208H           ;Weiter bis X >= 2 hoch 16

: X ist im SNG-Format bzw X >= 65536 wenn im DBL-Format

  1222 CD4F12           CALL    124FH           ;X solange durch 10 teilen bis
                                                ;X < 1E+6 bzw X < 1D+16

: X ist jetzt < 1E+6 bzw < 1E+16

  1225 E7               RST     20H             ;TSTTYP. Ist X im DBL-Format ?
  1226 300B             JR      NC,1233H        ;Ja: weiter bei 1233H

: X ist im SNG-Format

  1228 014391           LD      BC,9143H        ;BCDE = 1E+5
  122B 11F94F           LD      DE,4FF9H
  122E CD0C0A           CALL    0A0CH           ;CP X,BCDE = CP X,1E+5
  1231 1806             JR      1239H           ;weiter bei 1239H

: X ist im DBL-Format

  1233 116C13           LD      DE,136CH        ;DE : 1D+15
  1236 CD490A           CALL    0A49H           ;CP X,(DE) = CP X,1D+15

  1239 F24B12           JP      P,124BH         ;Fertig wenn
                                                ;X >= 1D+15 bzw. X >= 1E+5
  123C F1               POP     AF              ;Exponentenoffset zurück
  123D CD0B0F           CALL    0F0BH           ;X = X * 10 (SNG,DBL) A -1
  1240 F5               PUSH    AF              ;Exponentenoffset retten
  1241 18E2             JR      1225H           ;Weiter bis
                                                ;X >= 1E+5 bzw. X >= 1D+15
```

```
; X >= 1E+6 bzw. X >= 1D+16
; (Fortsetzung von 124FH ff)

  1243 F1           POP    AF          :Exponentenoffset zurück
  1244 CD180F       CALL   0F18H       :X = X/10, A +1
  1247 F5           PUSH   AF          :Exponentenoffset retten
  1248 CD4F12       CALL   124FH       :Nochmal testen

; Skalierung abgeschlossen:
;
;  1E+5 <= X < 1E+6 wenn X im SNG-Format
; 1E+15 <= X < 1E+16 wenn X im DBL-Format

  124B F1           POP    AF          :Exponentenoffset zurück
  124C B7           OR     A           :CY = 0 (für 11B5H ff)
  124D D1           POP    DE -        :DE zurück
  124E C9           RET

; X solange durch 10 teilen bis X < 1E+6 bzw. X < 1D+16

  124F E7           RST    20H         :TSTTYP. Ist X im DBL-Format ?
  1250 EA5E12       JP     PE,125EH    :Ja: weiter bei 125EH

; X ist im SNG-Format

  1253 017494       LD     BC,9474H    :BCDE = 1E+6
  1256 11F823       LD     DE,23F8H
  1259 CD0C0A       CALL   0A0CH       :CP X,BCDE = CP X,1E+6
  125C 1806         JR     1264H       ;weiter bei 1264H

; X ist im DBL-Format

  125E 117413       LD     DE,1374H    :DE : 1D+16
  1261 CD490A       CALL   0A49H       :CP X,(DE) = CP X,1D+16

  1264 E1           POP    HL          :HL = Ret-Adr.
  1265 F24312       JP     P,1243H     ;weiter bei 1243H wenn
                                       ;X >= 1E+6 bzw. X >= 1D+16
  1268 E9           JP     (HL)        :RET

; A Nullen in Buffer ab (HL) schreiben
; (Nachkommastellen)

  1269 B7           OR     A           :Zähler = 0 ?
  126A C8           RET    Z           :Ja: Fertig
  126B 3D           DEC    A           :Nein: Zähler -1
  126C 3630         LD     (HL),30H    ;'0' einsetzen
  126E 23           INC    HL          :Zeiger +1
  126F 18F9         JR     126AH       :Fertig ?
```

```
; A Nullen in Buffer ab (HL) schreiben und ',' und '.' setzen
; (Vorkommastellen)

  1271 2004          JR     NZ,1277H      :Zähler = 0 ?
                                          :Nein: weiter bei 1277H
  1273 C8            RET    Z             :Ja: Fertig
  1274 CD9112        CALL   1291H         :',' und '.' setzen
  1277 3630          LD     (HL),30H      :'0' einsetzen
  1279 23            INC    HL            :Zeiger +1
  127A 3D            DEC    A             :Zähler -1
  127B 18F6          JR     1273H         :Zähler = 0 ?

; Dezimalpunktposition und Zähler für die Tausendertrennung ermitteln

  127D 7B            LD     A,E           :A = Exponentenoffset
  127E 82            ADD    A,D           :+ maximale Anzahl der erzeugten
                                          :Stellen
  127F 3C            INC    A             :+ 1
  1280 47            LD     B,A           := Dezimalpunktposition
                                          :(= Anzahl der Vorkommastellen)
                                          :Zähler für die Tausendertrennung
                                          :ermitteln:
  1281 3C            INC    A             :A +1
  1282 D603          SUB    03H           :A = A div 3
  1284 30FC          JR     NC,1282H      :(Ganzzahlige Division)
  1286 C605          ADD    A,05H         :+ 5
  1288 4F            LD     C,A           :Ergibt Zähler für die Tausender-
                                          :trennung
  1289 3AD840        LD     A,(40D8H)     :A = Formatbyte
  128C E640          AND    40H           :Tausendertrennung erwünscht ?
  128E C0            RET    NZ            :Ja: B und C ok
  128F 4F            LD     C,A           :Nein: C auf 0 setzen
  1290 C9            RET

; ',' und '.' setzen
; B = Anzahl der restlichen Vorkommastellen (bis zum Dezimalpunkt)
; C = Anzahl der restlichen Ziffern bis zur nächsten Tausenderposition
; C = 0 wenn keine Tausendertrennung durchgeführt werden soll

  1291 05            DEC    B             :Vorkommastellen -1
                                          :Dezimalpunktposition erreicht ?
  1292 2008          JR     NZ,129CH      :Nein: ',' setzen

  1294 362E          LD     (HL),2EH      :Ja: '.' in den Buffer setzen

; Bufferzeiger der Dezimalpunktstelle retten
; Keine Tausendertrennung mehr durchführen

  1296 22F340        LD     (40F3H),HL    :und Bufferzeiger retten
  1299 23            INC    HL            :Bufferzeiger +1
  129A 48            LD     C,B           :C = 0 (Keine Tausendertrennung
                                          :mehr)
  129B C9            RET
```

```
; Dezimalpunkt noch nicht erreicht: Tausendertrennung einfügen

  129C 0D              DEC     C            ;Nächste Tausenderposition
                                            ;erreicht ?
  129D C0              RET     NZ           ;Nein: Fertig
  129E 362C            LD      (HL),2CH     ;Ja: ',' in den Buffer setzen
  12A0 23              INC     HL           ;Bufferzeiger +1
  12A1 0E03            LD      C,03H        ;Zähler = 3 für nächste
                                            ;Tausenderstelle
  12A3 C9              RET

; Fließkommazahl in unformatierten String umwandeln

  12A4 D5              PUSH    DE           ;DE retten
  12A5 E7              RST     20H          ;TSTTYP, Ist X im SNG-Format ?
  12A6 E2EA12          JP      PO,12EAH     ;Ja: weiter bei 12EAH

; X ist im DBL-Format (und 1E+15 <= X < 1E+16 !)
; Unformatierten String mit 17 Stellen erzeugen
; (10 Stellen mit Hilfe der DBL-Mantissen, 2 Stellen mit SNG-Mantissen
;  und 5 Stellen im INT-Format)

  12A9 C5              PUSH    BC           ;Stellenzähler retten
  12AA E5              PUSH    HL           ;Bufferzeiger retten
  12AB CDFC09          CALL    09FCH        ;Y = X
  12AE 217C13          LD      HL,137CH     ;HL : 0.5 (DBL)
  12B1 CDF709          CALL    09F7H        ;X = (HL) = 0.5
  12B4 CD770C          CALL    0C77H        ;X = X + Y = 0.5 + Y
                                            ;(X aufrunden)
  12B7 AF              XOR     A            ;CY = 0
  12B8 CD7B0B          CALL    0B7BH        ;Nachkommastellen löschen
  12BB E1              POP     HL           ;Bufferzeiger zurück
  12BC C1              POP     BC           ;Stellenzähler zurück
  12BD 118C13          LD      DE,138CH     ;DE : DBL-Mantissen
  12C0 3E0A            LD      A,0AH        ;Ab (DE) stehen 10 DBL-Mantissen

  12C2 CD9112          CALL    1291H        ;'.' und ',' setzen
  12C5 C5              PUSH    BC           ;Stellenzähler retten
  12C6 F5              PUSH    AF           ;Mantissenzähler retten
  12C7 E5              PUSH    HL           ;Bufferzeiger retten
  12C8 D5              PUSH    DE           ;Mantissenzeiger retten
  12C9 062F            LD      B,2FH        ;B = ASCII-Wert von '0' - 1

  12CB 04              INC     B            ;nächste Ziffer
  12CC E1              POP     HL           ;HL = Mantissenzeiger
  12CD E5              PUSH    HL           ;Mantissenzeiger retten
  12CE CD480D          CALL    0D48H        ;X = X - (HL): Mantisse
                                            ;subtrahieren, Unterlauf ?
  12D1 30F8            JR      NC,12CBH     ;Nein: Nächste Ziffer
```

```
12D3 E1         POP   HL          ;Ja: Mantissenzeiger zurück
12D4 CD360D     CALL  0D36H       ;X = X + (HL) (Subtraktion
                                  ;zurücknehmen)
12D7 EB         EX    DE,HL       ;DE = Mantissenzeiger
12D8 E1         POP   HL          ;HL = Bufferzeiger
12D9 70         LD    (HL),B      ;Ziffer einfügen
12DA 23         INC   HL          ;Bufferzeiger auf nächste
                                  ;Dezimalstellen erhöhen
12DB F1         POP   AF          ;Mantissenzähler zurück
12DC C1         POP   BC          ;Stellenzähler zurück
12DD 3D         DEC   A           ;Mantissenzähler -1
12DE 20E2       JR    NZ,12C2H    ;Nächste Dezimalstelle

12E0 C5         PUSH  BC          ;Stellenzähler retten
12E1 E5         PUSH  HL          ;Bufferzeiger retten
12E2 211D41     LD    HL,411DH    ;HL = Zeiger auf X (DBL)
12E5 CDB109     CALL  09B1H       ;X = BCDE = (HL) (SNG)
                                  ;Die restlichen LSBs der DBL-Zahl
                                  ;nach X als SNG-Zahl schieben
12E8 180C       JR    12F6H       ;Die restlichen Dezimalstellen
                                  ;werden im SNG-Format verarbeitet
                                  ;(da X jetzt < 1D+6 ist)

; X ist im SNG-Format (und 1E+5 <= X < 1E+6 !)
; Unformatierten String mit 7 Stellen erzeugen
; (2 Stellen mit Hilfe der SNG-Mantissen und die restlichen 5 Stellen
;  im INT-Format)

12EA C5         PUSH  BC          ;Stellenzähler retten
12EB E5         PUSH  HL          ;Bufferzeiger retten
12EC CD0807     CALL  0708H       ;X = X + 0.5 (Mantisse richtig-
                                  ;stellen)
12EF 3C         INC   A           ;A <> 0 (für 0AFBH)
12F0 CDFB0A     CALL  0AFBH       ;Alle Nachkommastellen löschen
12F3 CDB409     CALL  09B4H       ;X = BCDE
                                  ;(BCDE war Ergebnis von 0AFBH)
12F6 E1         POP   HL          ;Bufferzeiger zurück
12F7 C1         POP   BC          ;Stellenzähler zurück
12F8 AF         XOR   A           ;CY = 0
12F9 11D213     LD    DE,13D2H    ;DE = Mantissenzeiger

12FC 3F         CCF               ;CY = 1 beim ersten Durchlauf
                                  ;danach CY = 0
12FD CD9112     CALL  1291H       ;',' und '.' setzen
1300 C5         PUSH  BC          ;Stellenzähler retten
1301 F5         PUSH  AF          ;Wiederholungsflag retten
1302 E5         PUSH  HL          ;Bufferzeiger retten
1303 D5         PUSH  DE          ;Mantissenzeiger retten
1304 CDBF09     CALL  09BFH       ;BCDE = X
```

```
1307 E1          POP    HL          ;Mantissenzeiger zurück nach HL
1308 062F        LD     B,2FH       ;A = ASCII-Wert von '0' - 1

130A 04          INC    B           ;nächste Ziffer
130B 7B          LD     A,E         ;CDE = CDE - (HL):
130C 96          SUB    (HL)        ;(Mantissen subtrahieren)
130D 5F          LD     E,A
130E 23          INC    HL
130F 7A          LD     A,D
1310 9E          SBC    A,(HL)
1311 57          LD     D,A
1312 23          INC    HL
1313 79          LD     A,C
1314 9E          SBC    A,(HL)
1315 4F          LD     C,A
1316 2B          DEC    HL          ;Mantissenzeiger wieder auf den
1317 2B          DEC    HL          ;Anfang der Mantisse zeigen
                                    ;lassen
                                    ;Unterlauf bei Mantissensub-
                                    ;traktion ?
1318 30F0        JR     NC,130AH    ;Nein: nächste Ziffer

131A CDB707      CALL   07B7H       ;Ja: CDE = CDE + (HL)
                                    ;(Subtraktion zurücknehmen)
131D 23          INC    HL          ;Mantissenzeiger +1
131E CDB409      CALL   09B4H       ;X = BCDE, neuen Wert nach X
                                    ;zurückschreiben
1321 EB          EX     DE,HL       ;DE = Mantissenzeiger
1322 E1          POP    HL          ;Bufferzeiger zurück
1323 70          LD     (HL),B      ;Ziffer einsetzen
1324 23          INC    HL          ;Bufferzeiger +1
1325 F1          POP    AF          ;Wiederholungsflag zurück
1326 C1          POP    BC          ;Stellenzähler zurück
1327 38D3        JR     C,12FCH     ;Wiederholung ? (Es gibt nur
                                    ;2 Mantissen im SNG-Format)

1329 13          INC    DE          ;Mantissenzeiger +2
132A 13          INC    DE          ;(Zur INT-Verarbeitung)
132B 3E04        LD     A,04H       ;Nur noch 4 Mantissen verarbeiten
                                    ;(Die erste INT-Mantisse mit
                                    ;10000 steht auch im SNG-Format)
132D 1806        JR     1335H       ;Restliche Ziffern im INT-Format
                                    ;verarbeiten
```

```
: X ist im INT-Format (und 0 <= X < 32768 !)
: Unformatierten String mit 5 Stellen erzeugen

132F D5              PUSH   DE            ;DE retten
1330 11D813          LD     DE,13D8H      ;DE = Mantissenzeiger
1333 3E05            LD     A,05H         ;5 Mantissen

1335 CD9112          CALL   1291H         ;',' und '.' setzen
1338 C5              PUSH   BC            ;Stellenzähler retten
1339 F5              PUSH   AF            ;Mantissenzähler retten
133A E5              PUSH   HL            ;Bufferzeiger retten
133B EB              EX     DE,HL         ;HL = Mantissenzeiger
133C 4E              LD     C,(HL)        ;BC = Mantisse
133D 23              INC    HL
133E 46              LD     B,(HL)
133F C5              PUSH   BC            ;Mantisse retten
1340 23              INC    HL            ;HL = Zeiger auf nächste Mantisse
1341 E3              EX     (SP),HL       ;Mantissenzeiger retten
                                          ;Mantisse zurück nach HL
1342 EB              EX     DE,HL         ;DE = Mantisse
1343 2A2141          LD     HL,(4121H)    ;HL = X
1346 062F            LD     B,2FH         ;B = ASCII-Wert von '0' - 1

1348 04              INC    B             ;nächste Ziffer
1349 7D              LD     A,L           ;HL = HL -DE
134A 93              SUB    E             ;Mantisse subtrahieren
134B 6F              LD     L,A
134C 7C              LD     A,H
134D 9A              SBC    A,D
134E 67              LD     H,A           ;Unterlauf ?
134F 30F7            JR     NC,1348H      ;Nein: nächste Ziffer
1351 19              ADD    HL,DE         ;Ja: Subtraktion zurücknehmen
1352 222141          LD     (4121H),HL    ;und neuen Wert nach X zurück-
                                          ;schreiben
1355 D1              POP    DE            ;Mantissenzeiger zurück
1356 E1              POP    HL            ;Bufferzeiger zurück
1357 70              LD     (HL),B        ;Ziffer einsetzen
1358 23              INC    HL            ;Bufferzeiger +1
1359 F1              POP    AF            ;Mantissenzähler zurück
135A C1              POP    BC            ;Stellenzähler zurück
135B 3D              DEC    A             ;Mantissenzähler -1
135C 20D7            JR     NZ,1335H      ;Nächste Mantisse
135E CD9112          CALL   1291H         ;',' und '.' setzen
1361 77              LD     (HL),A        ;String mit 00H abschließen
1362 D1              POP    DE            ;DE zurück
1363 C9              RET
```

: Fließkommakonstanten

```
1364 00                                   ;1D+10
1365 00
1366 00
1367 00
1368 F9                                   ;1E+10
1369 02
136A 15
136B A2

136C FD                                   ;1D+15
136D FF
136E 9F
136F 31
1370 A9                                 - ;1E+15
1371 5F
1372 63
1373 B2

1374 FE                                   ;1D+16
1375 FF
1376 03
1377 BF
1378 C9                                   ;1E+16
1379 1B
137A 0E
137B B6

137C 00                                   ;0.5 (DBL)
137D 00
137E 00
137F 00
1380 00                                   ;0.5 (SNG)
1381 00
1382 00
1383 80

1384 00                                   ;1D+16
1385 00
1386 04
1387 BF
1388 C9                                   ;1E+16
1389 1B
138A 0E
138B B6
```

```
: Festkommakonstanten (Mantissen für Zahlenumwandlung)
:       LSB                 MSB

  138C 00 80 C6 A4 7E 8D 03              :1000000000000000 = 1D+15

  1393 00 40 7A 10 F3 5A 00              :100000000000000  = 1D+14

  139A 00 A0 72 4E 18 09 00              :10000000000000   = 1D+13

  13A1 00 10 A5 D4 E8 00 00              :1000000000000    = 1D+12

  13A8 00 E8 76 48 17 00 00              :100000000000     = 1D+11

  13AF 00 E4 0B 54 02 00 00              :10000000000      = 1D+10

  13B6 00 CA 9A 3B 00 00 00              :1000000000       = 1D+9

  13BD 00 E1 F5 05 00 00 00              :100000000        = 1D+8

  13C4 80 96 98 00 00 00 00              :10000000         = 1D+7

  13CB 40 42 0F 00 00 00 00              :1000000          = 1D+6

  13D2 A0 86 01                          :100000           = 1E+5

  13D5 10 27 00                          :10000            = 1E+4

  13D8 10 27                             :10000            (INT)

  13DA E8 03                             :1000

  13DC 64 00                             :100

  13DE 0A 00                             :10

  13E0 01 00                             :1
```

```
; UPRO für SQR und ATN: Ergebnis der Routine negieren (X = -X)

  13E2 218209         LD      HL,0982H     ;HL = Adresse der Routine X = -X
  13E5 E3             EX      (SP),HL      ;HL als RET-Adresse im Stack
                                           ;ablegen
  13E6 E9             JP      (HL)         ;Zurück nach SQR bzw. ATN

; X = SQR ( X ) = X hoch 0.5

  13E7 CDA409         CALL    09A4H        ;(SP) = X
  13EA 218013         LD      HL,1380H     ;HL : Konstante 0.5
  13ED CDB109         CALL    09B1H        ;X = BCDE = (HL)
  13F0 1803           JR      13F5H        ;(SP) hoch X ausrechnen

; X = (SP) hoch X

  13F2 CDB10A         CALL    0AB1H        ;X = CSNG (X) (Exponent)
  13F5 C1             POP     BC           ;BCDE = (SP) (Basis)
  13F6 D1             POP     DE
  13F7 CD5509         CALL    0955H        ;TEST2: Ist der Exponent Null ?
  13FA 78             LD      A,B          ;A = Exp (Basis)
  13FB 283C           JR      Z,1439H      ;Ja: EXP (0) ausrechnen
                                           ;(Ergibt auch 1)
  13FD F20414         JP      P,1404H      ;Ist der Exponent positiv ?
                                           ;Ja: weiter bei 1404H
  1400 B7             OR      A            ;Nein: Ist die Basis Null ?
                                           ;(Bei negativem Exponenten)
  1401 CA9A19         JP      Z,199AH      ;Ja: /0-Error
  1404 B7             OR      A            ;Nein: Ist die Basis Null ?
                                           ;(Bei positivem Exponenten)
  1405 CA7907         JP      Z,0779H      ;Ja: Ergebnis ist Null
  1408 D5             PUSH    DE           ;Basis retten
  1409 C5             PUSH    BC
  140A 79             LD      A,C          ;A = MSB (Basis)
  140B F67F           OR      7FH          ;Vorzeichen der Basis überprüfen
  140D CDBF09         CALL    09BFH        ;BCDE = X = Exponent
  1410 F22114         JP      P,1421H      ;weiter bei 1421H wenn Basis
                                           ;positiv (CY = 0, Z = 0 !)
  1413 D5             PUSH    DE           ;Exponent retten
  1414 C5             PUSH    BC
  1415 CD400B         CALL    0B40H        ;X = INT (X) = INT (Exponent)
  1418 C1             POP     BC           ;Exponent zurück
  1419 D1             POP     DE
  141A F5             PUSH    AF           ;LSB (X) retten
  141B CD0C0A         CALL    0A0CH        ;CP X,BCDE
                                           ;CP INT(Exponent), Exponent
                                           ;Ist der Exponent ganz-
                                           ;zahlig ?
  141E E1             POP     HL           ;LSB (X) zurück
  141F 7C             LD      A,H          ;A = LSB (X) = INT (Exponent),
                                           ;da Exponent < 88 (also im
                                           ;Bereich 0 - 255)
  1420 1F             RRA                  ;CY = A.0 (Niedrigstes Bit des
                                           ;Exponenten)
                                           ;Ist der Exponent ungerade ?
```

```
1421 E1                  POP     HL          ;X = (SP) = Basis
1422 222341              LD      (4123H),HL
1425 E1                  POP     HL
1426 222141              LD      (4121H),HL
1429 DCE213              CALL    C,13E2H     ;Ja: Ergebnis negieren
                                             ;(wenn bei negativer Basis der
                                             ;Exponent ungerade ist)
142C CC8209              CALL    Z,0982H     ;Basis positiv machen
                                             ;(wenn der Exponent ganzzahlig
                                             ;ist, siehe 141BH)
142F D5                  PUSH    DE          ;Exponent retten
1430 C5                  PUSH    BC
1431 CD0908              CALL    0809H       ;X=LOG(X) (Basis logarithmieren)
1434 C1                  POP     BC          ;Exponent zurück
1435 D1                  POP     DE
1436 CD4708              CALL    0847H       ;X = BCDE * X
                                             ;= Exponent * LOG (Basis)
                                             ;Und nun EXP(Exponent*LOG(Basis))
                                             ;berechnen

; X = EXP ( X )
; Nur für -88.7228 <= X <= 87.3365 berechenbar

1439 CDA409              CALL    09A4H       ;(SP) = X, Argument retten
143C 013881              LD      BC,8138H    ;BCDE = 1.4427 = 1 / LOG(2)
143F 113BAA              LD      DE,0AA3BH
1442 CD4708              CALL    0847H       ;X = BCDE * Arg = Arg / LOG(2)
                                             ;(= 2-Exponent des Ergebnisses !)
1445 3A2441              LD      A,(4124H)   ;A = Exp (X)
1448 FE88                CP      88H         ;Ist Exp(X) >= 88H ?
                                             ;-> Ist X >= 2 hoch 8 ?
                                             ;-> Ist Arg/LOG(2) > 127 ?
                                             ;(bzw. Ist Arg/LOG(2) < -128 ?
                                             ;-> Ist Arg > 88.0297
                                             ;(bzw. Ist Arg < -88.7228 ?)
144A D23109              JP      NC,0931H    ;Ja: OV-Error wenn Arg > 88.0297
                                             ;Erg = 0 wenn X < -88.7228
144D CD400B              CALL    0B40H       ;A = X = INT(X)
                                             ;( X = INT( Arg/LOG(2) )
                                             ;( = 2-Exponent des Ergebnisses )
1450 C680                ADD     A,80H       ;A > 7DH ? (A + 82H > FFH ?)
1452 C602                ADD     A,02H       ;-> INT(Arg/LOG(2)) > 125 ?
                                             ;-> X > 87.3365 ?
1454 DA3109              JP      C,0931H     ;Ja: OV-Error
1457 F5                  PUSH    AF          ;2-Exponent + Offset (80H) + 2
                                             ;retten
1458 21F807              LD      HL,07F8H    ;HL : 1
145B CD0B07              CALL    070BH       ;X = X + (HL) = X + 1
145E CD4108              CALL    0841H       ;X = X * LOG(2)
1461 F1                  POP     AF          ;2-Exponent zurück
1462 C1                  POP     BC          ;BCDE = (SP) = Argument
1463 D1                  POP     DE
```

```
 1464 F5              PUSH   AF          :2-Exponent retten
 1465 CD1307          CALL   0713H       :X = BCDE - X
 1468 CD8209          CALL   0982H       :X = -X
 146B 217914          LD     HL,1479H    :HL : Koeffizienten
 146E CDA914          CALL   14A9H       :Reihe2 berechnen
 1471 110000          LD     DE,0000H    :DE = 0000H
 1474 C1              POP    BC          :B = 2-Exponent + 2
 1475 4A              LD     C,D         :C = 00H
 1476 C34708          JP     0847H       :X = BCDE * X
                                         := (2 hoch 2+2-Exponent) * X

; Koeffizienten für EXP
; ' ! ' bedeutet hier die Fakultät der Zahl ( 3! = 1 * 2 * 3 )

 1479 08                                 :8 Koeffizienten

 147A 40                                 :-1.41316E-4 = -1/7076
 147B 2E                                 :          ca. 1/5040 = -1/7!
 147C 94
 147D 74

 147E 70                                 :1.32988E-3 = 1/752
 147F 4F                                 :          ca. 1/720 = 1/6!
 1480 2E
 1481 77

 1482 6E                                 :-8.30136E-3 = -1/120
 1483 02                                 :              = -1/5!
 1484 88
 1485 7A

 1486 E6                                 :0.0416574 = 1/24
 1487 A0                                 :            = 1/4!
 1488 2A
 1489 7C

 148A 50                                 :-0.166665 = -1/6
 148B AA                                 :            = -1/3!
 148C AA
 148D 7E

 148E FF                                 :0.5 = 1/2
 148F FF                                 :    = 1/2!
 1490 7F
 1491 7F

 1492 00                                 :-1 = -1/1!
 1493 00
 1494 80
 1495 81

 1496 00                                 :1
 1497 00
 1498 00
 1499 81
```

```
; Reihenberechnung 1
; Berechnet die Taylor-Reihe der Form:
; y = k1*x + k2*x*x*x + k3*x*x*x*x*x ... (k1, k2, k3 sind Koeffizienten)
; I: HL = Zeiger auf die Koeffiziententabelle
;         Das erste Byte der Tabelle gibt die Anzahl der Koeffizienten an,
;         dann folgen die Koeffizienten in umgekehrter Reihenfolge (k1 zuletzt)
;    X  = Faktor in den Reihengliedern (x im Beispiel)
; O: X  = Ergebnis der Reihenberechnung (y im Beispiel)

  149A CDA409        CALL   09A4H      ;(SP) = X
  149D 11320C        LD     DE,0C32H   ;DE = Adresse für X = X * (SP)
  14A0 D5            PUSH   DE         ;Als RET-Adr im Stack ablegen
  14A1 E5            PUSH   HL         ;Tabellenzeiger retten
  14A2 CDBF09        CALL   09BFH      ;BCDE = X
  14A5 CD4708        CALL   0847H      ;X = X * BCDE = X * X
  14A8 E1            POP    HL  -      ;Zeiger zurück und
                                       ;Reihenberechnung 2 mit X*X
                                       ;durchführen und Ergebnis
                                       ;nochmal mit X multiplizieren

; Reihenberechnung 2
; Berechnet die Taylor-Reihe der Form:
; y = k1 + k2*x + k3*x*x + k4*x*x*x ...
; I und O siehe Reihenberechnung 1

  14A9 CDA409        CALL   09A4H      ;(SP) = X
  14AC 7E            LD     A,(HL)     ;A = Anzahl der Koeffizienten
  14AD 23            INC    HL         ;HL = 1. Zahl (letzter Koeffi-
                                       ;zient in der Reihe)
  14AE CDB109        CALL   09B1H      ;X = BCDE = (HL) = 1. Zahl
  14B1 06F1          LD     B,0F1H     ;--
 *14B2   F1          POP    AF         ;Zähler zurück
  14B3 C1            POP    BC         ;BCDE = X
  14B4 D1            POP    DE
  14B5 3D            DEC    A          ;Noch Koeffizienten ?
  14B6 C8            RET    Z          ;Nein: Fertig
  14B7 D5            PUSH   DE         ;BCDE retten
  14B8 C5            PUSH   BC
  14B9 F5            PUSH   AF         ;Zähler retten
  14BA E5            PUSH   HL         ;Zeiger retten
  14BB CD4708        CALL   0847H      ;Ergebnis = Ergebnis * BCDE
  14BE E1            POP    HL         ;Zeiger zurück
  14BF CDC209        CALL   09C2H      ;BCDE = (HL) (= Koeffizient)
  14C2 E5            PUSH   HL         ;Zeiger retten
  14C3 CD1607        CALL   0716H      ;Ergebnis = Ergebnis + BCDE
  14C6 E1            POP    HL         ;Zeiger zurück
  14C7 18E9          JR     14B2H      ;Nächstes Glied berechnen
```

```
; X = RND ( X )
; Für X >= 1 gilt: RND ( X ) = INT( RND (0) * INT(X) + 1 )

  14C9 CD7F0A            CALL    0A7FH        ;X = HL = CINT (Arg)
  14CC 7C                LD      A,H          ;Ist das Argument negativ ?
  14CD B7                OR      A
  14CE FA4A1E            JP      M,1E4AH      ;Ja: FC-Error
  14D1 B5                OR      L            ;Ist das Argument gleich Null ?
  14D2 CAF014            JP      Z,14F0H      ;Ja: RND (0) berechnen
  14D5 E5                PUSH    HL           ;Argument retten
  14D6 CDF014            CALL    14F0H        ;RND (0) berechnen
  14D9 CDBF09            CALL    09BFH        ;BCDE = X = RND (0)
  14DC EB                EX      DE,HL        ;(SP) = BCDE, HL = Argument
  14DD E3                EX      (SP),HL
  14DE C5                PUSH    BC
  14DF CDCF0A            CALL    0ACFH        ;X = CSNG (HL)
  14E2 C1                POP     BC           ;BCDE = (SP) = RND (0)
  14E3 D1                POP     DE
  14E4 CD4708            CALL    0847H        ;X = X * BCDE = Arg * RND (0)
  14E7 21F807            LD      HL,07F8H     ;HL : 1
  14EA CD0B07            CALL    070BH        ;X = X + (HL) = X + 1
  14ED C3400B            JP      0B40H        ;X = CINT (X)

; X = RND ( 0 )
;   = letzte Zufallszahl * 0.253514 + 0.022228 (ohne Beachtung des Übertrags)

  14F0 219040            LD      HL,4090H     ;HL : Multiplikator
  14F3 E5                PUSH    HL           ;Zeiger retten
  14F4 110000            LD      DE,0000H     ;CDE = 000000H
  14F7 4B                LD      C,E          ;(In CDE wird das Ergebnis
                                              ;berechnet)
  14F8 2603              LD      H,03H        ;H = Bytezähler
                                              ;(3 Bytes Mantisse)
  14FA 2E08              LD      L,08H        ;L = Bitzähler
                                              ;(8 Bits pro Byte)
  14FC EB                EX      DE,HL        ;CDE um ein Bit nach links
  14FD 29                ADD     HL,HL        ;schieben
  14FE EB                EX      DE,HL        ;(für Multiplikation)
  14FF 79                LD      A,C
  1500 17                RLA
  1501 4F                LD      C,A
  1502 E3                EX      (SP),HL      ;Zähler retten, Zeiger zurück
  1503 7E                LD      A,(HL)       ;Nächstes Bit des Multiplikators
  1504 07                RLCA                 ;ins CY-Flag schieben
  1505 77                LD      (HL),A       ;Nächstes Bit = 1 ?
  1506 E3                EX      (SP),HL      ;Zeiger retten, Zähler zurück
  1507 D21615            JP      NC,1516H     ;Nein: weiter bei 1516H
  150A E5                PUSH    HL           ;Ja: Zähler retten
  150B 2AAA40            LD      HL,(40AAH)   ;HL = LSBs der letzten Zufalls-
                                              ;zahl
  150E 19                ADD     HL,DE        ;Zum Ergebnis addieren
  150F EB                EX      DE,HL
```

```
1510 3AAC40        LD      A,(40ACH)      ;A = MSB der letzten Zufallszahl
1513 89            ADC     A,C            ;Zum Ergebnis addieren
1514 4F            LD      C,A
1515 E1            POP     HL             ;Zähler zurück
1516 2D            DEC     L              ;Bitzähler -1
                                          ;8 Bits verarbeitet ?
1517 C2FC14        JP      NZ,14FCH       ;Nein: nächstes Bit
151A E3            EX      (SP),HL        ;Ja: Zähler retten, Zeiger zurück
151B 23            INC     HL             ;Zeiger +1 (nächstes Byte des
                                          ;Multiplikators verarbeiten)
151C E3            EX      (SP),HL        ;Zeiger retten, Zähler zurück
151D 25            DEC     H              ;Bytezähler -1
                                          ;3 Bytes verarbeitet ?
151E C2FA14        JP      NZ,14FAH       ;Nein: nächstes Byte
1521 E1            POP     HL             ;Zeiger vom Stack löschen
1522 2165B0        LD      HL,0B065H      ;und 05B065H (= 0.022228)
1525 19            ADD     HL,DE          ;zur Mantisse addieren
1526 22AA40        LD      (40AAH),HL     ;Neue Zufallszahl abspeichern
1529 CDEF0A        CALL    0AEFH          ;VT auf SNG-Format setzen
152C 3E05          LD      A,05H          ;MSB verrechnen
152E 89            ADC     A,C
152F 32AC40        LD      (40ACH),A      ;und abspeichern
1532 EB            EX      DE,HL          ;CDE = Mantisse der neuen
                                          ;Zufallszahl
1533 0680          LD      B,80H          ;Exp auf 80H setzen
1535 212541        LD      HL,4125H       ;HL : Signflag
1538 70            LD      (HL),B         ;Signflag = 80H
                                          ;(Bei SFLOAT eine positive Zahl
                                          ;erzeugen)
1539 2B            DEC     HL             ;HL : Exp (X)
153A 70            LD      (HL),B         ;Exp (X) = 80H (Ergebnis < 1)
153B 4F            LD      C,A            ;MSB in BCDE einsetzen
153C 0600          LD      B,00H          ;LSB von CDEB = 00H
153E C36507        JP      0765H          ;CDEB ins SNG-Format umwandeln
                                          ;und in X ablegen (SFLOAT)

: X = COS ( X ) = SIN ( X + PI/2 )
: PI = Kreiszahl (3.14159)

1541 218B15        LD      HL,158BH       ;HL : PI/2
1544 CD0B07        CALL    070BH          ;X = Arg + PI/2
```

```
: X = SIN ( X )

  1547 CDA409        CALL   09A4H        :(SP) = Arg
  154A 014983        LD     BC,8349H     :BCDE = 6.28319 = 2*PI
  154D 11DB0F        LD     DE,0FDBH
                                         :X in den Breich von -1 bis +1
                                         :skalieren:
  1550 CDB409        CALL   09B4H        :X = BCDE = 2*PI
  1553 C1            POP    BC           :BCDE = Arg
  1554 D1            POP    DE
  1555 CDA208        CALL   08A2H        :X = X/BCDE = Arg/(2*PI)
  1558 CDA409        CALL   09A4H        :(SP) = X = Arg/(2*PI)
  155B CD400B        CALL   0B40H        :X = INT (X) = INT( Arg/(2*PI) )
  155E C1            POP    BC           :BCDE = Arg/(2*PI)
  155F D1            POP    DE
  1560 CD1307        CALL   0713H        :X = BCDE - X
                                         := Arg/(2*PI) - INT(Arg/(2*PI))
                                         :(Vielfache von 2PI ausblenden)
                                         :Das Argument (bzw X) liegt jetzt
                                         :im Bereich zwischen -1 (ent-
                                         :spricht -2*PI) und +1 (ent-
                                         :spricht +2*PI)
  1563 218F15        LD     HL,158FH     :HL : 1/4
  1566 CD1007        CALL   0710H        :X = (HL) - X = 1/4 - X
  1569 CD5509        CALL   0955H        :TEST2, X >= 0 ? (Arg > PI/2) ?
  156C 37            SCF                 :CY = 1
  156D F27715        JP     P,1577H      :Ja: weiter bei 1577H
  1570 CD0807        CALL   0708H        :Nein:X = X + 1/2
  1573 CD5509        CALL   0955H        :TEST2, X >= 0 ? (Arg < PI/2) ?
  1576 B7            OR     A            :CY = 0 (für 1582H)
  1577 F5            PUSH   AF           :Flags retten
  1578 F48209        CALL   P,0982H      :Ja: X = -X
  157B 218F15        LD     HL,158FH     :HL = 1/4
  157E CD0B07        CALL   070BH        :X = X + (HL) = X + 1/4
  1581 F1            POP    AF           :Flags zurück
  1582 D48209        CALL   NC,0982H     :X = -X wenn 1/4 - X > 0 war
  1585 219315        LD     HL,1593H     :HL = Zeiger auf Koeffizienten
  1588 C39A14        JP     149AH        :Reihe1 berechnen

  158B DB                                :Konstante 1.5708 = PI/2
  158C 0F
  158D 49
  158E 81

  158F 00                                :Konstante 0.25 = 1/4
  1590 00
  1591 00
  1592 7F
```

```
; Koeffiziententabelle für SIN und COS
; ' ! ' bedeutet Fakultät der Zahl

  1593 05                                    ;5 Koeffizienten

  1594 BA                                    ;39.7107
  1595 D7                                    ;= ((2*PI) hoch 9) / 9!
  1596 1E
  1597 86

  1598 64                                    ;-76.575
  1599 26                                    ;= - ((2*PI) hoch 7) / 7!
  159A 99
  159B 87

  159C 58                                    ;81.6022
  159D 34                                    ;= ((2*PI) hoch 5) / 5!
  159E 23
  159F 87

  15A0 E0                                    ;-41.3417
  15A1 5D                                    ;= - ((2*PI) hoch 3) / 3!
  15A2 A5
  15A3 86

  15A4 DA                                    ;6.28319 = 2*PI
  15A5 0F                                    ;= ((2*PI) hoch 1) / 1!
  15A6 49
  15A7 83
```

```
: X = TAN ( X ) = SIN (X) / COS (X)

  15A8 CDA409        CALL   09A4H        :(SP) = Arg
  15AB CD4715        CALL   1547H        :X = SIN (Arg)
  15AE C1            POP    BC           :BCHL = Arg
  15AF E1            POP    HL
  15B0 CDA409        CALL   09A4H        :(SP) = X = SIN (Arg)
  15B3 EB            EX     DE,HL        :BCDE = Arg
  15B4 CDB409        CALL   09B4H        :X = BCDE = Arg
  15B7 CD4115        CALL   1541H        :X = COS (Arg)
  15BA C3A008        JP     08A0H        :Erg = (SP) / X
                                         := SIN (Arg) / COS (Arg)

: X = ATN ( X )

  15BD CD5509        CALL   0955H        :TEST2. Argument < 0 ?
  15C0 FCE213        CALL   M,13E2H      :Ja: Ergebnis nachher negieren
  15C3 FC8209        CALL   M,0982H      :und mit positivem Argument
                                         :weiterrechnen
  15C6 3A2441        LD     A,(4124H)    :A = Exp (Arg)
  15C9 FE81          CP     81H          :Argument < 1 ?
  15CB 380C          JR     C,15D9H      :Ja: weiter bei 15D9H
  15CD 010081        LD     BC,8100H     :BCDE = 1.0
  15D0 51            LD     D,C
  15D1 59            LD     E,C
  15D2 CDA208        CALL   08A2H        :X = BCDE / X = 1/Arg
  15D5 211007        LD     HL,0710H     :HL = Adresse für X = (HL) - X
  15D8 E5            PUSH   HL           :Als RET-Adresse im Stack ablegen
  15D9 21E315        LD     HL,15E3H     :HL : Koeffiziententabelle
  15DC CD9A14        CALL   149AH        :Reihe1 berechnen
  15DF 218B15        LD     HL,158BH     :HL : PI/2
  15E2 C9            RET                 :Erg = PI/2 - X wenn das
                                         :Argument > 1 war
                                         :(Denn die Reihenberechnung
                                         :stimmt nur für -1 < X < +1)
```

```
; Koeffizienten für ATN

  15E3 09                                            :9 Koeffizienten

  15E4 4A                                            :2.86623E-3 ca. 1/349
  15E5 D7
  15E6 3B
  15E7 78

  15E8 02                                            :-0.0161657 ca. 1/62
  15E9 6E
  15EA 84
  15EB 7B

  15EC FE                                            :0.0429096 ca 1/23
  15ED C1
  15EE 2F
  15EF 7C

  15F0 74                                            :-0.0752896 ca. -1/11
  15F1 31
  15F2 9A
  15F3 7D

  15F4 84                                            :0.106563 ca. 1/9
  15F5 3D
  15F6 5A
  15F7 7D

  15F8 C8                                            :-0.142089 ca. -1/7
  15F9 7F
  15FA 91
  15FB 7E

  15FC E4                                            :0.199936 = 1/5
  15FD BB
  15FE 4C
  15FF 7E

  1600 6C                                            :-0.333331 = -1/3
  1601 AA
  1602 AA
  1603 7F

  1604 00                                            :1.0 = 1/1
  1605 00
  1606 00
  1607 81
```

; Adressentabelle der BASIC-Funktionen (Token D7H bis FAH)

```
1608 8A 09                 ;SGN     098AH
160A 37 0B                 ;INT     0B37H
160C 77 09                 ;ABS     0977H
160E D4 27                 ;FRE     27D4H
1610 EF 2A                 ;INP     2AEFH
1612 F5 27                 ;POS     27F5H
1614 E7 13                 ;SQR     13E7H
1616 C9 14                 ;RND     14C9H
1618 09 08                 ;LOG     0809H
161A 39 14                 ;EXP     1439H
161C 41 15                 ;COS     1541H
161E 47 15                 ;SIN     1547H
1620 A8 15                 ;TAN     15A8H
1622 BD 15                 ;ATN     15BDH
1624 AA 2C                 ;PEEK    2CAAH
1626 52 41                 ;CVI     4152H
1628 58 41                 ;CVS     4158H
162A 5E 41                 ;CVD     415EH
162C 61 41                 ;EOF     4161H
162E 64 41                 ;LOC     4164H
1630 67 41                 ;LOF     4167H
1632 6A 41                 ;MKI$    416AH
1634 6D 41                 ;MKS$    416DH
1636 70 41                 ;MKD$    4170H
1638 7F 0A                 ;CINT    0A7FH
163A B1 0A                 ;CSNG    0AB1H
163C DB 0A                 ;CDBL    0ADBH
163E 26 0B                 ;FIX     0B26H
1640 03 2A                 ;LEN     2A03H
1642 36 28                 ;STR$    2836H
1644 C5 2A                 ;VAL     2AC5H
1646 0F 2A                 ;ASC     2A0FH
1648 1F 2A                 ;CHR$    2A1FH
164A 61 2A                 ;LEFT$   2A61H
164C 91 2A                 ;RIGHT$  2A91H
164E 9A 2A                 ;MID$    2A9AH
```

```
: Tabelle der BASIC-Keywords in der Reihenfolge ihrer Token-Werte
: Zur Trennung zwischen den Keywords ist jeweils das höchste Bit
: des ersten Zeichens auf '1' gesetzt.

  1650 C5 4E 44                                  :END
  1653 C6 4F 52                                  :FOR
  1656 D2 45 53 45 54                            :RESET
  165B D3 45 54                                  :SET
  165E C3 4C 53                                  :CLS
  1661 C3 4D 44                                  :CMD
  1664 D2 41 4E 44 4F 4D                         :RANDOM
  166A CE 45 58 54                               :NEXT
  166E C4 41 54 41                               :DATA
  1672 C9 4E 50 55 54                            :INPUT
  1677 C4 49 4D                                  :DIM
  167A D2 45 41 44                               :READ
  167E CC 45 54                                  :LET
  1681 C7 4F 54 4F                               :GOTO
  1685 D2 55 4E                                  :RUN
  1688 C9 46                                     :IF
  168A D2 45 53 54 4F 52 45                      :RESTORE
  1691 C7 4F 53 55 42                            :GOSUB
  1696 D2 45 54 55 52 4E                         :RETURN
  169C D2 45 4D                                  :REM
  169F D3 54 4F 50                               :STOP
  16A3 C5 4C 53 45                               :ELSE
  16A7 D4 52 4F 4E                               :TRON
  16AB D4 52 4F 46 46                            :TROFF
  16B0 C4 45 46 53 54 52                         :DEFSTR
  16B6 C4 45 46 49 4E 54                         :DEFINT
  16BC C4 45 46 53 4E 47                         :DEFSNG
  16C2 C4 45 46 44 42 4C                         :DEFDBL
  16C8 CC 49 4E 45                               :LINE
  16CC C5 44 49 54                               :EDIT
  16D0 C5 52 52 4F 52                            :ERROR
  16D5 D2 45 53 55 4D 45                         :RESUME
  16DB CF 55 54                                  :OUT
  16DE CF 4E                                     :ON
  16E0 CF 50 45 4E                               :OPEN
  16E4 C6 49 45 4C 44                            :FIELD
  16E9 C7 45 54                                  :GET
  16EC D0 55 54                                  :PUT
  16EF C3 4C 4F 53 45                            :CLOSE
  16F4 CC 4F 41 44                               :LOAD
```

```
16F8 CD 45 52 47 45                    :MERGE
16FD CE 41 4D 45                       :NAME
1701 CB 49 4C 4C                       :KILL
1705 CC 53 45 54                       :LSET
1709 D2 53 45 54                       :RSET
170D D3 41 56 45                       :SAVE
1711 D3 59 53 54 45 4D                 :SYSTEM
1717 CC 50 52 49 4E 54                 :LPRINT
171D C4 45 46                          :DEF
1720 D0 4F 4B 45                       :POKE
1724 D0 52 49 4E 54                    :PRINT
1729 C3 4F 4E 54                       :CONT
172D CC 49 53 54                       :LIST
1731 CC 4C 49 53 54                    :LLIST
1736 C4 45 4C 45 54 45                 :DELETE
173C C1 55 54 4F                       :AUTO
1740 C3 4C 45 41 52                    :CLEAR
1745 C3 4C 4F 41 44                    :CLOAD
174A C3 53 41 56 45                    :CSAVE
174F CE 45 57                          :NEW
1752 D4 41 42 28                       :TAB(
1756 D4 4F                             :TO
1758 C6 4E                             :FN
175A D5 53 49 4E 47                    :USING
175F D6 41 52 50 54 52                 :VARPTR
1765 D5 53 52                          :USR
1768 C5 52 4C                          :ERL
176B C5 52 52                          :ERR
176E D3 54 52 49 4E 47 24              :STRING$
1775 C9 4E 53 54 52                    :INSTR
177A C3 48 45 43 4B                    :CHECK
177F D4 49 4D 45 24                    :TIME$
1784 CD 45 4D                          :MEM
1787 C9 4E 4B 45 59 24                 :INKEY$
178D D4 48 45 4E                       :THEN
1791 CE 4F 54                          :NOT
1794 D3 54 45 50                       :STEP
1798 AB                                :+
1799 AD                                :-
179A AA                                :*
179B AF                                :/
179C DB                                :'Exponent'
179D C1 4E 44                          :AND
```

```
17A0 CF 52                       ;OR
17A2 BE                          ;>
17A3 BD                          ;=
17A4 BC                          ;<
17A5 D3 47 4E                    ;SGN
17A8 C9 4E 54                    ;INT
17AB C1 42 53                    ;ABS
17AE C6 52 45                    ;FRE
17B1 C9 4E 50                    ;INP
17B4 D0 4F 53                    ;POS
17B7 D3 51 52                    ;SQR
17BA D2 4E 44                    ;RND
17BD CC 4F 47                    ;LOG
17C0 C5 58 50                    ;EXP
17C3 C3 4F 53                    ;COS
17C6 D3 49 4E                    ;SIN
17C9 D4 41 4E                    ;TAN
17CC C1 54 4E                    ;ATN
17CF D0 45 45 4B                 ;PEEK
17D3 C3 56 49                    ;CVI
17D6 C3 56 53                    ;CVS
17D9 C3 56 44                    ;CVD
17DC C5 4F 46                    ;EOF
17DF CC 4F 43                    ;LOC
17E2 CC 4F 46                    ;LOF
17E5 CD 4B 49 24                 ;MKI$
17E9 CD 4B 53 24                 ;MKS$
17ED CD 4B 44 24                 ;MKD$
17F1 C3 49 4E 54                 ;CINT
17F5 C3 53 4E 47                 ;CSNG
17F9 C3 44 42 4C                 ;CDBL
17FD C6 49 58                    ;FIX
1800 CC 45 4E                    ;LEN
1803 D3 54 52 24                 ;STR$
1807 D6 41 4C                    ;VAL
180A C1 53 43                    ;ASC
180D C3 48 52 24                 ;CHR$
1811 CC 45 46 54 24              ;LEFT$
1816 D2 49 47 48 54 24           ;RIGHT$
181C CD 49 44 24                 ;MID$
1820 A7                          ;'
1821 80                          ;Ende der Tabelle
```

```
: Adressentabelle der BASIC-Befehle (Token von 80H bis BBH)

    1822 AE 1D                                              :END      1DAEH
    1824 A1 1C                                              :FOR      1CA1H
    1826 38 01                                              :RESET    0138H
    1828 35 01                                              :SET      0135H
    182A C9 01                                              :CLS      01C9H
    182C 73 41                                              :CMD      4173H
    182E D3 01                                              :RANDOM   01D3H
    1830 B6 22                                              :NEXT     22B6H
    1832 05 1F                                              :DATA     1F05H
    1834 9A 21                                              :INPUT    219AH
    1836 08 26                                              :DIM      2608H
    1838 EF 21                                              :READ     21EFH
    183A 21 1F                                              :LET      1F21H
    183C C2 1E                                              :GOTO     1EC2H
    183E A3 1E                                              :RUN      1EA3H
    1840 39 20                                              :IF       2039H
    1842 91 1D                                              :RESTORE  1D91H
    1844 B1 1E                                              :GOSUB    1EB1H
    1846 DE 1E                                              :RETURN   1EDEH
    1848 07 1F                                              :REM      1F07H
    184A A9 1D                                              :STOP     1DA9H
    184C 07 1F                                              :ELSE     1F07H
    184E F7 1D                                              :TRON     1DF7H
    1850 F8 1D                                              :TROFF    1DF8H
    1852 00 1E                                              :DEFSTR   1E00H
    1854 03 1E                                              :DEFINT   1E03H
    1856 06 1E                                              :DEFSNG   1E06H
    1858 09 1E                                              :DEFDBL   1E09H
    185A A3 41                                              :LINE     41A3H
    185C 60 2E                                              :EDIT     2E60H
    185E F4 1F                                              :ERROR    1FF4H
    1860 AF 1F                                              :RESUME   1FAFH
    1862 FB 2A                                              :OUT      2AFBH
    1864 6C 1F                                              :ON       1F6CH
    1866 79 41                                              :OPEN     4179H
    1868 7C 41                                              :FIELD    417CH
    186A 7F 41                                              :GET      417FH
    186C 82 41                                              :PUT      4182H
    186E 85 41                                              :CLOSE    4185H
    1870 88 41                                              :LOAD     4188H
    1872 8B 41                                              :MERGE    418BH
    1874 8E 41                                              :NAME     418EH
    1876 91 41                                              :KILL     4191H
    1878 97 41                                              :LSET     4197H
    187A 9A 41                                              :RSET     419AH
    187C A0 41                                              :SAVE     41A0H
    187E B2 02                                              :SYSTEM   02B2H
    1880 67 20                                              :LPRINT   2067H
    1882 5B 41                                              :DEF      415BH
    1884 B1 2C                                              :POKE     2CB1H
    1886 6F 20                                              :PRINT    206FH
    1888 E4 1D                                              :CONT     1DE4H
    188A 2E 2B                                              :LIST     2B2EH
```

```
  188C 29 2B                                 :LLIST    2B29H
  188E C6 2B                                 :DELETE   2BC6H
  1890 08 20                                 :AUTO     2008H
  1892 7A 1E                                 :CLEAR    1E7AH
  1894 1F 2C                                 :CLOAD    2C1FH
  1896 F5 2B                                 :CSAVE    2BF5H
  1898 49 1B                                 :NEW      1B49H

: Prioritätstabelle für Operatoren
: Höchster Wert entspricht höchster Priorität

  189A 79                                    :+
  189B 79                                    :-
  189C 7C                                    :*
  189D 7C                                    :/
  189E 7F                                    :Exponent
  189F 50                                    :AND
  18A0 46                                    :OR

: Adressentabelle für Typumwandlung

  18A1 DB 0A                                 :0ADBH : CDBL
  18A3 00 00                                 :0000H : ?
  18A5 7F 0A                                 :0A7FH : CINT
  18A7 F4 0A                                 :0AF4H : TM-Error wenn kein
                                             :        String in X
  18A9 B1 0A                                 :0AB1H : CSNG

: Adressentabelle für die vier Grundrechenarten und den Vergleich

: 1. Doppelte Genauigkeit

  18AB 77 0C                                 :0C77H : + (X = X + Y)
  18AD 70 0C                                 :0C70H : - (X = X - Y)
  18AF A1 0D                                 :0DA1H : * (X = X * Y)
  18B1 E5 0D                                 :0DE5H : / (X = X / Y)
  18B3 78 0A                                 :0A78H : Vergleich (CP X , Y)

: 2. Einfache Genauigkeit

  18B5 16 07                                 :0716H : + (X = BCDE + X)
  18B7 13 07                                 :0713H : - (X = BCDE - X)
  18B9 47 08                                 :0847H : * (X = BCDE * X)
  18BB A2 08                                 :08A2H : / (X = BCDE / X)
  18BD 0C 0A                                 :0A0CH : Vergleich (CP X , BCDE)

: 3. Integer

  18BF D2 0B                                 :0BD2H : + (X = DE + HL)
  18C1 C7 0B                                 :0BC7H : - (X = DE - HL)
  18C3 F2 0B                                 :0BF2H : * (X = DE * HL)
  18C5 90 24                                 :2490H : / (X = DE / HL)
  18C7 39 0A                                 :0A39H : Vergleich (CP HL , DE)
```

```
: Tabelle der Fehlermeldungen, nach Fehlercodes sortiert

   18C9 4E 46                                           :NF
   18CB 53 4E                                           :SN
   18CD 52 47                                           :RG
   18CF 4F 44                                           :OD
   18D1 46 43                                           :FC
   18D3 4F 56                                           :OV
   18D5 4F 4D                                           :OM
   18D7 55 4C                                           :UL
   18D9 42 53                                           :BS
   18DB 44 44                                           :DD
   18DD 2F 30                                           :/0
   18DF 49 44                                           :ID
   18E1 54 4D                                           :TM
   18E3 4F 53                                  -        :OS
   18E5 4C 53                                           :LS
   18E7 53 54                                           :ST
   18E9 43 4E                                           :CN
   18EB 4E 52                                           :NR
   18ED 52 57                                           :RW
   18EF 55 45                                           :UE
   18F1 4D 4F                                           :MO
   18F3 46 44                                           :FD
   18F5 53 4E                                           :SN

: Dieser Teil wird von Start 4 ins RAM ab 4080H kopiert
: (Erläuterungen siehe RAM-Listing)

   18F7 D600                    SUB     00H
   18F9 6F                      LD      L,A
   18FA 7C                      LD      A,H
   18FB DE00                    SBC     A,00H
   18FD 67                      LD      H,A
   18FE 78                      LD      A,B
   18FF DE00                    SBC     A,00H
   1901 47                      LD      B,A
   1902 3E00                    LD      A,00H
   1904 C9                      RET
   1905 4A 1E
   1907 40 E6 4D
   190A DB00                    IN      A,(00H)
   190C C9                      RET
   190D D300                    OUT     (00H),A
   190F C9                      RET
   1910 00
   1911 00
   1912 00
   1913 00
   1914 281E
   1916 00
   1917 4C
   1918 43
   1919 FEFF
   191B 0148
```

```
: Texte

  191C 20                                       :' Error'
  191E 45
  191F 72
  1920 72
  1921 6F
  1922 72
  1923 00

  1924 20                                       :' in'
  1925 69
  1926 6E
  1927 20
  1928 00

  1929 52                                       :'READY'
  192A 45
  192B 41
  192C 44
  192D 59
  192E 0D
  192F 00

  1930 42                                       :'Break'
  1931 72
  1932 65
  1933 61
  1934 6B
  1935 00
```

```
; UPRO für FOR, NEXT und RETURN
; Holt Daten vom Stack zurück
; I: DE = VARPTR der neuen Schleifenvariablen wenn eine neue FOR-TO-Schleife
;         begonnen wird
;    DE = VARPTR der bei NEXT angegebenen Variablen
;    DE = 0000H wenn bei NEXT keine Variable angegeben wurde
; O: DE = unverändert
;    HL = 'Stackzeiger' auf FOR-TO-Stack + 1 (wenn Z = 0)
;    HL = 'Stackzeiger' auf FOR-TO-Stack + 3 (wenn Z = 1)
;    Z  = 0 wenn kein FOR-TO-Stack gefunden wurde oder die Variable noch nicht
;           in einer Schleife gebraucht wird
;    Z  = 1 wenn der Aufruf von NEXT kam oder die Variable bereits in einer
;           Schleife gebraucht wird

 1936 210400        LD      HL,0004H
 1939 39            ADD     HL,SP       ;HL = SP + 4
                                        ;HL ist jetzt Stackzeiger wie
                                        ;nach zweimaligem POP
 193A 7E            LD      A,(HL)      ;Markierung vom Stack holen
 193B 23            INC     HL          ;'Stackzeiger' +1
 193C FE81          CP      81H         ;FOR-Markierung gefunden ?
 193E C0            RET     NZ          ;Nein: Fertig
                                        ;Ja: FOR-TO-Stack bearbeiten
 193F 4E            LD      C,(HL)      ;BC mit dem VARPTR der
                                        ;Schleifenvariablen laden
 1940 23            INC     HL
 1941 46            LD      B,(HL)
 1942 23            INC     HL
 1943 E5            PUSH    HL          ;'Stackzeiger' retten
 1944 69            LD      L,C         ;HL = VARPTR der Schleifen-
 1945 60            LD      H,B         ;variablen
 1946 7A            LD      A,D         ;Ist DE = 0000H ?
 1947 B3            OR      E           ;(Kam der Aufruf von NEXT ?)
 1948 EB            EX      DE,HL       ;DE = VARPTR der im Stack
                                        ;gefundenen (aktiven)
                                        ;Schleifenvariablen
                                        ;HL = VARPTR der neuen
                                        ;Schleifenvariablen
 1949 2802          JR      Z,194DH     ;Ja: Fertig
 194B EB            EX      DE,HL       ;DE und HL vertauschen
 194C DF            RST     18H         ;Beide VARPTR vergleichen
 194D 010E00        LD      BC,000EH    ;BC = Offset zum nächsten
                                        ;FOR-TO-Stack
 1950 E1            POP     HL          ;'Stackzeiger' zurück
 1951 C8            RET     Z           ;Fertig wenn der gesuchte VARPTR
                                        ;gefunden wurde
 1952 09            ADD     HL,BC       ;'Stackzeiger' zum nächsten
                                        ;FOR-TO-Stack erhöhen. (Es werden
                                        ;pro FOR-TO-Schleife 17 Bytes im
                                        ;Stack benötigt, HL wurde bereits
                                        ;in 193BH, 1940H und 1942H um 3
                                        ;erhöht plus 0EH (14) macht 17)
 1953 18E5          JR      193AH       ;Weiter im Stack suchen
```

```
; Programmtextverschiebung zum Einfügen einer neuen Zeile
; Kopiert Speicher von (BC) nach (HL) bis BC = DE ist
; I: BC = Zeiger auf altes Programmende (vor Verschiebung)
;    DE = ZP auf neue Zeile
;    HL = Zeiger auf neues Programmende (nach Verschiebung)
; O: DE = ZP auf neue Zeile
;    HL = DE

 1955 CD6C19        CALL   196CH        ;Ist noch Platz im Speicher ?
 1958 C5            PUSH   BC           ;BC und HL vertauschen
 1959 E3            EX     (SP),HL      ;(wegen RST 18H)
 195A C1            POP    BC
 195B DF            RST    18H          ;An neuer Zeile angekommen ?
 195C 7E            LD     A,(HL)       ;Zeichen vom alten Platz
 195D 02            LD     (BC),A       ;zum neuen Platz kopieren
 195E C8            RET    Z            ;Ja: Fertig
 195F 0B            DEC    BC           ;Zeiger -1
 1960 2B            DEC    HL
 1961 18F8          JR     195BH        ;Nächstes Zeichen kopieren

; Prüfung ob noch Platz im Speicher ist
; OM-Error wenn weniger als 2 * C Bytes noch vorhanden sind
; I: C = Anzahl der benötigten Bytes

 1963 E5            PUSH   HL           ;PTZ retten
 1964 2AFD40        LD     HL,(40FDH)   ;HL = Zeiger auf Beginn des
                                        ;freien Speichers
 1967 0600          LD     B,00H        ;BC = Anzahl der benötigten Bytes
 1969 09            ADD    HL,BC        ;HL = HL + 2*BC
 196A 09            ADD    HL,BC
 196B 3EE5          LD     A,0E5H       ;--

; Ist ab HL noch genügend Platz im Speicher ?
; I: HL = Zeiger auf freien Speicherplatz

*196C   E5          PUSH   HL           ;HL retten
                                        ;HL zeigt jetzt auf den neuen
                                        ;Anfang des freien Speichers
 196D 3EC6          LD     A,0C6H       ;HL = FFC6H - HL
 196F 95            SUB    L
 1970 6F            LD     L,A
 1971 3EFF          LD     A,0FFH
 1973 9C            SBC    A,H
 1974 3804          JR     C,197AH      ;OM-Error wenn HL > FF6CH
 1976 67            LD     H,A          ;MSB nach H zurück
 1977 39            ADD    HL,SP        ;HL = SP + (FFC6H - HL)
 1978 E1            POP    HL           ;HL zurück
 1979 D8            RET    C            ;RET wenn Platz bis zum Stack
                                        ;sonst OM-Error

; OM-Error

 197A 1E0C          LD     E,0CH        ;E = Fehlercode
 197C 1924          JR     19A2H        ;Zur Fehlerroutine
```

```
; Beendigung eines Programms ohne 'END'

  197E 2AA240           LD      HL,(40A2H)    ;HL = Aktuelle ZN
  1981 7C               LD      A,H           ;Wurde ein Programm beendet ?
  1982 A5               AND     L
  1983 3C               INC     A             ;(Ist die ZN <> 65535 ?)
  1984 2808             JR      Z,198EH       ;Nein: Über 198EH nach END
                                              ;springen
  1986 3AF240           LD      A,(40F2H)     ;ON ERROR GOTO-Flag gesetzt ?
  1989 B7               OR      A
  198A 1E22             LD      E,22H         ;E = Fehlercode für NR-Error
  198C 2014             JR      NZ,19A2H      ;NR-Error wenn Flag gesetzt
  198E C3C11D           JP      1DC1H         ;Sonst nach END springen

; SN-Error in einer DATA-Zeile

  1991 2ADA40           LD      HL,(40DAH)    ;HL = DATA-ZN
  1994 22A240           LD      (40A2H),HL    ;Als aktuelle ZN ab-
                                              ;speichern

; SN-Error

  1997 1E02             LD      E,02H         ;E = Fehlercode
  1999 011E14           LD      BC,141EH      ;--

; /0-Error

 *199A   1E14           LD      E,14H         ;E = Fehlercode
  199C 011E00           LD      BC,001EH      ;--

; NF-Error

 *199D   1E00           LD      E,00H         ;E = Fehlercode
  199F 011E24           LD      BC,241EH      ;--

; RW-Error

 *19A0   1E24           LD      E,24H         ;E = Fehlercode

; Fehlerroutine.
; Zeigt Fehlercode und Zeilennummer an und bricht Programm ab.
; I: E = (Fehlercode - 1) * 2
; O: - (RÜckkehr ins Basic oder zum Fehlerbehandlungsprogramm wenn
; ON ERROR GOTO aktiv war.)

  19A2 2AA240           LD      HL,(40A2H)    ;Aktuelle ZN
  19A5 22EA40           LD      (40EAH),HL    ;als ERL
  19A8 22EC40           LD      (40ECH),HL    ;und '.' abspeichern
  19AB 01B419           LD      BC,19B4H      ;BC = RET-Adr
  19AE 2AE840           LD      HL,(40E8H)    ;HL = Letzter SP-Stand
  19B1 C39A1B           JP      1B9AH         ;Stack neu initialisieren
```

```
19B4 C1          POP    BC            ;Stack korrigieren
19B5 7B          LD     A,E           ;A = Fehlercode
19B6 4B          LD     C,E           ;C = Fehlercode
19B7 329A40      LD     (409AH),A     ;Fehlercode als ERR abspeichern
19BA 2AE640      LD     HL,(40E6H)    ;HL = PTZ vor dem Fehler
19BD 22EE40      LD     (40EEH),HL    ;PTZ für RESUME retten
19C0 EB          EX     DE,HL         ;DE = PTZ
19C1 2AEA40      LD     HL,(40EAH)    ;HL = ZN
19C4 7C          LD     A,H           ;Ist die ZN
19C5 A5          AND    L             ;gleich FFFFH (65535) ?
19C6 3C          INC    A
19C7 2807        JR     Z,19D0H       ;Ja: Fehler in der aktiven
                                      ;Befehlsebene. Weiter bei 19D0H
19C9 22F540      LD     (40F5H),HL    ;Nein: ZN für CONT retten
19CC EB          EX     DE,HL         ;HL = PTZ
19CD 22F740      LD     (40F7H),HL    ;PTZ für CONT retten
19D0 2AF040      LD     HL,(40F0H)    ;HL = ZN von ON ERROR GOTO
19D3 7C          LD     A,H           ;war ON ERROR GOTO aktiv ?
19D4 B5          OR     L             ;(ZN <> 0)
19D5 EB          EX     DE,HL         ;DE = ZN
19D6 21F240      LD     HL,40F2H      ;HL : ON ERROR GOTO-Flag
19D9 2808        JR     Z,19E3H       ;Nein: weiter bei 19E3H
19DB A6          AND    (HL)          ;Flag gesetzt ?
19DC 2005        JR     NZ,19E3H      ;Nein: weiter bei 19E3H
19DE 35          DEC    (HL)          ;Flag -1 (als Zähler für RESUME)
19DF EB          EX     DE,HL         ;HL = PTZ der ON ERROR GOTO-Zeile
19E0 C3361D      JP     1D36H         ;Programm bei (HL) fortsetzen

: Fehler anzeigen

19E3 AF          XOR    A             ;ON ERROR GOTO-Flag löschen
19E4 77          LD     (HL),A
19E5 59          LD     E,C           ;E = Fehlercode
19E6 CDF920      CALL   20F9H         ;Neue Zeile beginnen
19E9 21C918      LD     HL,18C9H      ;HL = Zeiger auf Fehlercodes
19EC CDA641      CALL   41A6H         ;DOS
19EF 57          LD     D,A           ;DE = Offset für Fehlertabelle
19F0 3E3F        LD     A,3FH         ;'?' ausgeben
19F2 CD2A03      CALL   032AH
19F5 19          ADD    HL,DE         ;HL = Zeiger auf Fehlercode
19F6 7E          LD     A,(HL)        ;Beide Zeichen ausgeben
19F7 CD2A03      CALL   032AH         ;1. Zeichen
19FA D7          RST    10H           ;A = 2. Zeichen
19FB CD2A03      CALL   032AH         ;2. Zeichen ausgeben
19FE 211D19      LD     HL,191DH      ;HL : Text ' Error'
1A01 E5          PUSH   HL            ;HL retten
1A02 2AEA40      LD     HL,(40EAH)    ;HL = ERL
1A05 E3          EX     (SP),HL       ;ERL retten und Textzeiger zurück
```

```
; Einsprung von STOP (siehe 1DDEH)

  1A06 CD7935          CALL    3579H         ;Text und Ton ausgeben
  1A09 E1              POP     HL            ;ERL zurück
  1A0A 11FEFF          LD      DE,0FFFEH     ;Kommt der Fehler von der
                                             ;MEM SIZE-Frage ?
                                             ;(Ist ERL = 65534 ?)
  1A0D DF              RST     18H           ;HL und DE vergleichen
  1A0E CA7406          JP      Z,0674H       ;Ja: Zurück zu Start 1
  1A11 7C              LD      A,H           ;War der Fehler im Programm ?
  1A12 A5              AND     L             ;(Ist ERL <> 65535 ?)
  1A13 3C              INC     A
  1A14 C4A70F          CALL    NZ,0FA7H      ;Ja: 'in' und ZN ausgeben
  1A17 3EC1            LD      A,0C1H        ;Zurück ins BASIC
 *1A18   C1            POP     BC            ;--

; Einsprung zum aktiven Befehlsmodus

  1A19 CD8B03          CALL    038BH         ;Druckerausgabe beenden, nächste
                                             ;Ausgabe auf Bildschirm
  1A1C CDAC41          CALL    41ACH         ;DOS
  1A1F 00              NOP                   ;--
  1A20 00              NOP
  1A21 00              NOP
  1A22 CDF920          CALL    20F9H         ;Bildschirmausgabe abschließen
  1A25 212919          LD      HL,1929H      ;HL : Text 'READY'
  1A28 CD9238          CALL    3892H         ;CRTC initialisieren (LGR) und
                                             ;Text ausgeben
  1A2B 3A9A40          LD      A,(409AH)     ;A = Letzter Fehlercode
  1A2E D602            SUB     02H           ;War es ein SN-Error ?
  1A30 CC532E          CALL    Z,2E53H       ;Ja: Nach EDIT springen
  1A33 21FFFF          LD      HL,0FFFFH     ;Aktuelle ZN auf 65535 setzen
  1A36 22A240          LD      (40A2H),HL
  1A39 3AE140          LD      A,(40E1H)     ;AUTO aktiv ?
  1A3C B7              OR      A
  1A3D 2837            JR      Z,1A76H       ;Nein: Zur normalen Befehls-
                                             ;eingabe springen

; AUTO bearbeiten

  1A3F 2AE240          LD      HL,(40E2H)    ;HL = AUTO-ZN
  1A42 E5              PUSH    HL            ;ZN retten
  1A43 CDAF0F          CALL    0FAFH         ;ZN ausgeben
  1A46 D1              POP     DE            ;ZN zurück nach DE
  1A47 D5              PUSH    DE            ;ZN wieder retten
  1A48 CD2C1B          CALL    1B2CH         ;ZN suchen, schon vorhanden ?
  1A4B 3E2A            LD      A,2AH         ;A = '*'
  1A4D 3802            JR      C,1A51H       ;Ja: '*' ausgeben
  1A4F 3E20            LD      A,20H         ;Nein: ' ' ausgeben
  1A51 CD2A03          CALL    032AH         ;Zeichen ausgeben
  1A54 CD6103          CALL    0361H         ;Zeileneingabe
```

```
1A57 D1            POP    DE           ;ZN zurück
1A58 3006          JR     NC,1A60H     ;Zeile übernehmen wenn nicht
                                       ;BREAK gedrückt wurde
1A5A AF            XOR    A            ;sonst A = 00H
1A5B 32E140        LD     (40E1H),A    ;AUTO abschalten
1A5E 18B9          JR     1A19H        ;Zurück zum aktiven Befehls-
                                       ;modus

; AUTO-Zeile übernehmen

1A60 2AE440        LD     HL,(40E4H)   ;HL = Abstand zur nächsten ZN
1A63 19            ADD    HL,DE        ;Nächste AUTO-ZN errechnen
1A64 38F4          JR     C,1A5AH      ;AUTO abbrechen wenn nächste
                                       ;ZN größer als 65535 wird
1A66 D5            PUSH   DE           ;Jetzige ZN retten
1A67 11F9FF        LD     DE,0FFF9H    ;DE = 65530
1A6A DF            RST    18H          ;Neue ZN mit 65530 vergleichen
1A6B D1            POP    DE           ;Jetzige ZN zurück
1A6C 30EC          JR     NC,1A5AH     ;AUTO abbrechen wenn nächste
                                       ;ZN größer als 65529 wird
1A6E 22E240        LD     (40E2H),HL   ;Nächste ZN abspeichern
1A71 F6FF          OR     0FFH         ;A <> 0 setzen
1A73 C3EB2F        JP     2FEBH        ;Zeile durch EDIT-Routine
                                       ;übernehmen

; Normale Befehls- bzw. Zeileneingabe im aktiven Befehlsmodus

1A76 3E3E          LD     A,3EH        ;A = '>'
1A78 CD2A03        CALL   032AH        ;Zeichen ausgeben
1A7B CD6103        CALL   0361H        ;Zeileneingabe
1A7E DA331A        JP     C,1A33H      ;Zurück zum aktiven Befehls-
                                       ;modus wenn BREAK gedrückt
                                       ;wurde
1A81 D7            RST    10H          ;A = 1. Zeichen der eingegebenen
                                       ;Zeile
1A82 3C            INC    A            ;A = 00H ?
1A83 3D            DEC    A            ;(Kein OR A damit CY nicht
                                       ;beeinflusst wird)
1A84 CA331A        JP     Z,1A33H      ;Zurück zum aktiven Befehlsmodus
                                       ;wenn nichts eingegeben wurde
1A87 F5            PUSH   AF           ;Flags retten
1A88 CD5A1E        CALL   1E5AH        ;Nummer decodieren
1A8B 2B            DEC    HL           ;HL auf letzte Ziffer zurück
1A8C 7E            LD     A,(HL)       ;(Es werden alle der Zeilen-
1A8D FE20          CP     20H          ;nummer folgenden Leerzeichen
1A8F 28FA          JR     Z,1A8BH      ;mit übernommen)
1A91 23            INC    HL           ;HL zeigt auf erstes Zeichen der
                                       ;Zeile
1A92 7E            LD     A,(HL)       ;Ist das erste Zeichen
1A93 FE20          CP     20H          ;ein Leerzeichen ?
1A95 CCC909        CALL   Z,09C9H      ;Ja: HL +1 (Das erste Leerzeichen
                                       ;wird nicht mit übernommen, da
                                       ;LIST ein Leerzeichen nach der
                                       ;Zeilennummer automatisch setzt)
```

```
1A98 D5          PUSH   DE          :ZN retten
1A99 CDC01B      CALL   1BC0H       :Zwischencode erzeugen
1A9C D1          POP    DE          :ZN zurück
1A9D F1          POP    AF          :Flags zurück
1A9E 22E640      LD     (40E6H),HL  :HL als aktuellen PTZ abspeichern
1AA1 CDB241      CALL   41B2H       :DOS
1AA4 D25A1D      JP     NC,1D5AH    :Zeile direkt ausführen, wenn
                                    :keine ZN angegeben wurde

; Zeile ins Programm übernehmen

1AA7 D5          PUSH   DE          :ZN retten
1AA8 C5          PUSH   BC          :Zeilenlänge retten
1AA9 AF          XOR    A           :A = 00H
1AAA 32DD40      LD     (40DDH),A   :STOP-Flag löschen
1AAD D7          RST    10H         :A = 1. Zeichen des Zwischencodes
1AAE B7          OR     A           :A = 00H ?
                                    :(Wurde nur eine Zeilennummer und
                                    :kein Befehlstext angegeben ?)
1AAF F5          PUSH   AF          :Flags retten
1AB0 EB          EX     DE,HL       :Zeiger retten
1AB1 22EC40      LD     (40ECH),HL  :ZN als '.'-ZN abspeichern
1AB4 EB          EX     DE,HL       :Zeiger zurück
1AB5 CD2C1B      CALL   1B2CH       :Ist die ZN schon im Programm
                                    :vorhanden ?
1AB8 C5          PUSH   BC          :ZP retten
1AB9 DCE42B      CALL   C,2BE4H     :Ja: Zeile löschen
1ABC D1          POP    DE          :ZP zurück nach DE
1ABD F1          POP    AF          :Flags zurück
                                    :Z = 1 ? (siehe 1AAEH)
1ABE D5          PUSH   DE          :ZP retten
1ABF 2827        JR     Z,1AE8H     :Ja: Die Zeile sollte nur ge-
                                    :löscht werden. Jetzt müssen nur
                                    :noch die ZPs im Programm
                                    :erneuert werden
1AC1 D1          POP    DE          :ZP zurück
1AC2 2AF940      LD     HL,(40F9H)  :HL : Programmende
1AC5 E3          EX     (SP),HL     :HL = Zeilenlänge
1AC6 C1          POP    BC          :BC : Programmende
1AC7 09          ADD    HL,BC       :HL : Neues Programmende
1AC8 E5          PUSH   HL          :HL retten
1AC9 CD5519      CALL   1955H       :Platz für neue Zeile schaffen
1ACC E1          POP    HL          :Zeiger auf neues Programmende
1ACD 22F940      LD     (40F9H),HL  :zurück und abspeichern
1AD0 EB          EX     DE,HL       :HL = neuer ZP
1AD1 74          LD     (HL),H      :ZP auf nächste Zeile <> 0 setzen
1AD2 D1          POP    DE          :ZN zurück
1AD3 E5          PUSH   HL          :ZP retten
1AD4 23          INC    HL          :ZP + 2
1AD5 23          INC    HL
1AD6 73          LD     (HL),E      :Neue ZN ins Programm setzen
1AD7 23          INC    HL
1AD8 72          LD     (HL),D
1AD9 23          INC    HL
```

```
1ADA EB             EX     DE,HL       ;DE = Zeiger auf freien Platz für
                                       ;den Zeilentext
1ADB 2AA740         LD     HL,(40A7H)  ;HL = Zeiger auf Zeilenbuffer
1ADE EB             EX     DE,HL       ;HL = Zeiger auf Programm
                                       ;DE = Zeiger auf neue Zeile +2
1ADF 1B             DEC    DE          ;DE -2
1AE0 1B             DEC    DE
1AE1 1A             LD     A,(DE)      ;Zeichen vom Buffer
1AE2 77             LD     (HL),A      ;ins Programm übertragen
1AE3 23             INC    HL          ;Zeiger +1
1AE4 13             INC    DE
1AE5 B7             OR     A           ;Zeilenende erreicht ?
1AE6 20F9           JR     NZ,1AE1H    ;Nein: Weitermachen
1AE8 D1             POP    DE          ;Ja: ZP der neuen Zeile nach DE

1AE9 CDFC1A         CALL   1AFCH       ;Ab DE alle ZPs im Programm
                                       ;erneuern
1AEC CDB541         CALL   41B5H       ;DOS
1AEF CD5D1B         CALL   1B5DH       ;CLEAR
1AF2 CDB841         CALL   41B8H       ;DOS
1AF5 C3331A         JP     1A33H       ;Zurück zum aktiven Befehls-
                                       ;modus

; Alle ZPs im Programm erneuern

1AF8 2AA440         LD     HL,(40A4H)  ;HL = Anfang des Programms
1AFB EB             EX     DE,HL       ;DE = Anfang des Programms

; Alle ZPs ab DE im Programm erneuern

1AFC 62             LD     H,D         ;HL = ZP
1AFD 6B             LD     L,E
1AFE 7E             LD     A,(HL)      ;ZP zur nächsten Zeile = 0 ?
1AFF 23             INC    HL
1B00 B6             OR     (HL)        ;(Programmende erreicht ?)
1B01 C8             RET    Z           ;Ja: Fertig
1B02 23             INC    HL          ;Nein: HL auf Zeilentext erhöhen
1B03 23             INC    HL
1B04 23             INC    HL
1B05 AF             XOR    A           ;A = 00H
1B06 BE             CP     (HL)        ;00H (Ende der Zeile) suchen
1B07 23             INC    HL          ;Zeiger +1
1B08 20FC           JR     NZ,1B06H    ;Weitersuchen
1B0A EB             EX     DE,HL       ;DE = ZP auf nächste Zeile
1B0B 73             LD     (HL),E      ;ZP auf nächste Zeile in aktuelle
1B0C 23             INC    HL          ;Zeile setzen
1B0D 72             LD     (HL),D
1B0E 18EC           JR     1AFCH       ;Nächste Zeile bearbeiten
```

```
; Zeilennummern bei LIST und DELETE decodieren
; (LIST. , LIST AA, LIST AA-EE, LIST AA-, LIST -EE etc.)
;
; I: HL   = PTZ auf Zeichen nach Befehl (Zeilennummer im ASCII-Format)
; O: BC   = Zeiger auf die erste Zeile (AA im Beispiel)
;    DE   = Anfangs-ZN (AA im Beispiel) (Default = 0)
;    HL   = Zeiger auf die nächste Zeile (nach AA)
;    (SP) = End-ZN (EE im Beispiel) (Default = 65529)

 1B10 110000          LD     DE,0000H     ;Default Anfangs-ZN = 0
 1B13 D5              PUSH   DE           ;Anfangs-ZN retten
 1B14 2809            JR     Z,1B1FH      ;weiter bei 1B1FH wenn keine ZNs
                                          ;angegeben wurden
 1B16 D1              POP    DE           ;Anfangs-ZN aus Stack löschen
 1B17 CD4F1E          CALL   1E4FH        ;Anfangs-ZN decodieren
 1B1A D5              PUSH   DE  -        ;Anfangs-ZN retten
                                          ;(= 0 wenn '-' angegeben)
 1B1B 280B            JR     Z,1B28H      ;weiter bei 1B28H wenn nur eine
                                          ;ZN angegeben wurde
 1B1D CF              RST    08H          ;'-' vorhanden ?
 1B1E CE              DEFB   '-'-Token
 1B1F 11FAFF          LD     DE,0FFFAH    ;Default End-ZN = 65529
 1B22 C44F1E          JP     Z,1E4FH      ;End-ZN decodieren
 1B25 C29719          JP     NZ,1997H     ;SN-Error ?
 1B28 EB              EX     DE,HL        ;HL = End-ZN
 1B29 D1              POP    DE           ;DE = Anfangs-ZN
 1B2A E3              EX     (SP),HL      ;End-ZN retten
 1B2B E5              PUSH   HL           ;RET-Adr. zurück ins Stack

; Suche Zeilennummer DE im Programm
; I: DE = Zeilennummer der gesuchten Zeile
; O: BC = Zeiger auf die gesuchte Zeile (falls gefunden)
;         oder Zeiger auf das Programmende (falls nicht gefunden)
;    DE = Zeilennummer
;    HL = Zeiger auf die nächste Zeile
;    Z = 1 und CY = 1 bei erfolgreicher Suche
;    Z = 1 und CY = 0 : BC zeigt auf Programmende
;    Z = 0 und CY = 0 : BC zeigt auf die größte Zeile mit
;                       einer ZN < der gesuchten ZN

 1B2C 2AA440          LD     HL,(40A4H)   ;HL = Zeiger auf erste Programm-
                                          ;zeile
 1B2F 44              LD     B,H          ;BC = ZP
 1B30 4D              LD     C,L
 1B31 7E              LD     A,(HL)       ;ZP auf nächsten Zeile = 0 ?
 1B32 23              INC    HL
 1B33 B6              OR     (HL)
 1B34 2B              DEC    HL           ;INC HL wieder aufheben
 1B35 C8              RET    Z            ;Ja: Fertig
 1B36 23              INC    HL           ;HL auf ZN erhöhen
 1B37 23              INC    HL
 1B38 7E              LD     A,(HL)       ;HL = Zeilennummer
 1B39 23              INC    HL
 1B3A 66              LD     H,(HL)
 1B3B 6F              LD     L,A
 1B3C DF              RST    18H          ;Gefundene ZN mit gesuchter
                                          ;ZN vergleichen
```

```
1B3D 60             LD      H,B           :ZP zurück nach HL
1B3E 69             LD      L,C
1B3F 7E             LD      A,(HL)        :HL = Zeiger auf nächste Zeile
1B40 23             INC     HL
1B41 66             LD      H,(HL)
1B42 6F             LD      L,A
1B43 3F             CCF                   :CY = 1 bei erfolgreichem
                                          :Vergleich
1B44 C8             RET     Z             :Fertig wenn Zeile gefunden
1B45 3F             CCF                   :CCF wieder zurücknehmen
1B46 D0             RET     NC            :RET wenn gefundene ZN größer als
                                          :die gesuchte ZN ist
1B47 18E6           JR      1B2FH         :Nächste Zeile überprüfen

: NEW

1B49 C0             RET     NZ            :SN-Error ?
1B4A CDC901         CALL    01C9H         :CLS
1B4D 2AA440         LD      HL,(40A4H)    :HL = Zeiger auf Programmstart
1B50 CDF81D         CALL    1DF8H         :TROFF
1B53 32E140         LD      (40E1H),A     :AUTO abschalten
1B56 77             LD      (HL),A        :ZP zur zweiten Zeile
1B57 23             INC     HL            :auf 0 setzen
1B58 77             LD      (HL),A
1B59 23             INC     HL
1B5A 22F940         LD      (40F9H),HL    :Programmende = Programmstart + 2

: Einsprung für RUN ohne Zeilennummer (RET auf Programmschleife)

1B5D 2AA440         LD      HL,(40A4H)    :HL = Programmstart
1B60 2B             DEC     HL            :HL -1

: CLEAR ohne Argument

1B61 22DF40         LD      (40DFH),HL    :PTZ zum ersten bzw nächsten
                                          :Befehl abspeichern
1B64 061A           LD      B,1AH         :DEFSNG A-Z (B = 26)
1B66 210141         LD      HL,4101H      :HL = Zeiger auf DEF-Tabelle
1B69 3604           LD      (HL),04H      :Typcode in Tabelle setzen
1B6B 23             INC     HL            :Zeiger +1
1B6C 10FB           DJNZ    1B69H         :Nächster 'Buchstabe'
1B6E AF             XOR     A             :A = 0
1B6F 32F240         LD      (40F2H),A     :ON ERROR GOTO-Flag löschen
1B72 6F             LD      L,A           :HL = 0000H
1B73 67             LD      H,A
1B74 22F040         LD      (40F0H),HL    :ON ERROR GOTO-ZN = 0
1B77 22F740         LD      (40F7H),HL    :CONT-PTZ = 0
1B7A 2AB140         LD      HL,(40B1H)    :HL = TOPMEM
1B7D 22D640         LD      (40D6H),HL    :Adresse des letzten Strings im
                                          :Stringspeichers = TOPMEM
                                          :(Alle Strings löschen)
```

```
1B80 CD911D        CALL   1D91H         ;RESTORE
1B83 2AF940        LD     HL,(40F9H)    ;HL = Zeiger auf Programmende
1B86 22FB40        LD     (40FBH),HL    ;Ende der Variablentabellen
1B89 22FD40        LD     (40FDH),HL    ;gleich dem Programmende setzen
                                        ;-> alle Variablen löschen
1B8C CDBB41        CALL   41BBH         ;DOS
1B8F C1            POP    BC            ;BC = RET-Adr
1B90 2AA040        LD     HL,(40A0H)    ;HL = Beginn des Stringspeichers
1B93 2B            DEC    HL            ;HL -2
1B94 2B            DEC    HL
1B95 22E840        LD     (40E8H),HL    ;Programmstack neu setzen
1B98 23            INC    HL
1B99 23            INC    HL
1B9A F9            LD     SP,HL         ;SP = Beginn des Stringspeichers
1B9B 21B540        LD     HL,40B5H      ;Stringtabelle zurücksetzen
1B9E 22B340        LD     (40B3H),HL
1BA1 CD8B03        CALL   038BH         ;Druckerausgabe abschließen
1BA4 CD6921        CALL   2169H         ;Nächste Ausgabe auf Bildschirm
1BA7 AF            XOR    A             ;A = 00H
1BA8 67            LD     H,A           ;HL = 0000H
1BA9 6F            LD     L,A
1BAA 32DC40        LD     (40DCH),A     ;Feldvariablen freigeben
1BAD E5            PUSH   HL            ;Stackende durch 0000H markieren
1BAE C5            PUSH   BC            ;RET-Adr zurück ins Stack
1BAF 2ADF40        LD     HL,(40DFH)    ;HL = PTZ auf ersten bzw nächsten
                                        ;Befehl
1BB2 C9            RET
```

: UPRO für INPUT
: Ausgabe von '? ' und Zeileneingabe

```
1BB3 3E3F          LD     A,3FH         ;A = '?'
1BB5 CD2A03        CALL   032AH         ;Zeichen ausgeben
1BB8 3E20          LD     A,20H         ;A = ' '
1BBA CD2A03        CALL   032AH         ;Zeichen ausgeben
1BBD C36103        JP     0361H         ;Zur Zeileneingabe springen
```

```
; UPRO für den aktiven Befehlsmodus
; Programmtext im Zeilenbuffer in Zwischencode umwandeln
; Textzeiger = Zeiger auf eingegebenen Text
; Bufferzeiger = Zeiger auf codierten Text
;
; I: HL = Textzeiger
; O: BC = Anzahl der im Programmspeicher benötigten Bytes (Zeilenlänge)
;    DE = Zeiger auf das Ende des codierten Textes
;    HL = Zeiger auf den Anfang des codierten Textes - 1

  1BC0 AF         XOR   A            ;A = 00H
  1BC1 32B040     LD    (40B0H),A    ;Textcodierung freigeben
  1BC4 4F         LD    C,A          ;Zähler = 0
  1BC5 EB         EX    DE,HL        ;DE = Textzeiger
  1BC6 2AA740     LD    HL,(40A7H)   ;HL = Anfang des Zeilenbuffers
  1BC9 2B         DEC   HL
  1BCA 2B         DEC   HL
  1BCB EB         EX    DE,HL        ;DE = Bufferzeiger
                                     ;HL = Textzeiger
  1BCC 7E         LD    A,(HL)       ;A = Textzeichen
  1BCD FE20       CP    20H          ;Zeichen = ' ' ?
  1BCF CA5B1C     JP    Z,1C5BH      ;Ja: Zeichen abspeichern
  1BD2 47         LD    B,A          ;B = Zeichen
  1BD3 FE22       CP    22H          ;Anfang eines Strings (' " ') ?
  1BD5 CA771C     JP    Z,1C77H      ;Ja: Alle folgenden Zeichen
                                     ;abspeichern bis zum Ausführungs-
                                     ;zeichen (Bis Zeichen = B)
  1BD8 B7         OR    A            ;Zeilenende erreicht ?
  1BD9 CA7D1C     JP    Z,1C7DH      ;Ja: weiter bei 1C7DH
  1BDC 3AB040     LD    A,(40B0H)    ;Codierung freigegeben ?
  1BDF B7         OR    A
  1BE0 7E         LD    A,(HL)       ;A = Textzeichen
  1BE1 C25B1C     JP    NZ,1C5BH     ;Nein: Zeichen übernehmen
  1BE4 FE3F       CP    3FH          ;Ja: Ist es ein '?' ?
  1BE6 3EB2       LD    A,0B2H       ;A = PRINT-Token
  1BE8 CA5B1C     JP    Z,1C5BH      ;Ja: PRINT-Token abspeichern
  1BEB 7E         LD    A,(HL)       ;A = Textzeichen
  1BEC FE30       CP    30H          ;Ist das Zeichen eine Zahl ?
  1BEE 3805       JR    C,1BF5H      ;Nein: Zeichen codieren
  1BF0 FE3C       CP    3CH          ;Ja: Ist das Zeichen kleiner '<' ?

  1BF2 DA5B1C     JP    C,1C5BH      ;Ja: Zeichen übernehmen
                                     ;(Die Zeichen '<', '=' und '>'
                                     ;werden codiert)
```

: Zeichen bzw Text codieren

```
1BF5 D5         PUSH  DE         ;Bufferzeiger retten
1BF6 114F16     LD    DE,164FH   ;DE = Zeiger auf Keywordtabelle
1BF9 C5         PUSH  BC         ;Zähler retten
1BFA 013D1C     LD    BC,1C3DH   ;RET-Adr auf 1C3DH setzen
1BFD C5         PUSH  BC
1BFE 067F       LD    B,7FH      ;B = Tokenzähler
1C00 7E         LD    A,(HL)     ;A = Textzeichen
1C01 FE61       CP    61H        ;Ist das Zeichen < 'a' ?
1C03 3807       JR    C,1C0CH    ;Ja: weiter bei 1C0CH
1C05 FE7B       CP    7BH        ;Ist das Zeichen > 'z' ?
1C07 3003       JR    NC,1C0CH   ;Ja: weiter bei 1C0CH
1C09 E65F       AND   5FH        ;Kleinschrift in Großschrift
                                 ;umwandeln
1C0B 77         LD    (HL),A     ;Zeichen wieder ablegen
1C0C 4E         LD    C,(HL)     ;C = Textzeichen
1C0D EB         EX    DE,HL      ;HL = Tabellenzeiger
1C0E 23         INC   HL         ;Tabellenzeiger +1
1C0F B6         OR    (HL)       ;Nächstes Keyword erreicht ?
1C10 F20E1C     JP    P,1C0EH    ;Nein: HL auf nächstes Keyword
                                 ;erhöhen
1C13 04         INC   B          ;Ja: Tokenzähler +1
1C14 7E         LD    A,(HL)     ;A = Tabellenzeichen
1C15 CDE238     CALL  38E2H      ;Colour-Basic Befehle abfangen
1C18 B9         CP    C          ;Textzeichen mit Tabellenzeichen
                                 ;vergleichen
1C19 20F3       JR    NZ,1C0EH   ;Ungleich: Tabellenzeiger zum
                                 ;nächsten Keyword erhöhen und
                                 ;neu vergleichen
1C1B EB         EX    DE,HL      ;DE = Tabellenzeiger
                                 ;HL = Bufferzeiger auf erstes
                                 ;Zeichen
1C1C E5         PUSH  HL         ;HL retten
1C1D 13         INC   DE         ;Nächstes Zeichen
1C1E 1A         LD    A,(DE)     ;aus Tabelle holen
1C1F B7         OR    A          ;Nächstes Keyword erreicht ?
                                 ;(War der Vergleich bei allen
                                 ;Zeichen des letzten Keywords
                                 ;erfolgreich ?)
1C20 FA391C     JP    M,1C39H    ;Ja: Token in B übernehmen
1C23 4F         LD    C,A        ;Nein: C = Tabellenzeichen
1C24 78         LD    A,B        ;A = Tokenzähler
1C25 FE8D       CP    8DH        ;Bei GOTO-Token ?
1C27 2002       JR    NZ,1C2BH   ;Nein: weiter bei 1C2BH
1C29 D7         RST   10H        ;Ja: Textzeiger zum nächsten
                                 ;Zeichen erhöhen (GOTO kann auch
                                 ;GO TO geschrieben werden !)
1C2A 2B         DEC   HL         ;HL -1 wegen HL +1 in RST 10H
```

```
1C2B 23            INC    HL          ;Textzeiger +1
1C2C 7E            LD     A,(HL)      ;A = nächstes Textzeichen
1C2D FE61          CP     61H         ;Kleinschrift ?
1C2F 3802          JR     C,1C33H     ;Nein: OK
1C31 E65F          AND    5FH         ;Ja: In Groß umwandeln
1C33 B9            CP     C           ;Textzeichen und Tabellenzeichen
                                      ;identisch ?
1C34 28E7          JR     Z,1C1DH     ;Ja: nächstes Zeichen vergleichen
1C36 E1            POP    HL          ;Nein: Textzeiger zurück auf
                                      ;erstes Zechen
1C37 18D3          JR     1C0CH       ;und mit nächstem Keyword
                                      ;vergleichen

; Vergleich erfolgreich abgeschlossen

1C39 48            LD     C,B         ;C = Token
1C3A F1            POP    AF          ;Textzeiger vom Stack löschen
1C3B EB            EX     DE,HL       ;DE = Textzeiger
1C3C C9            RET                ;RET nach 1C3DH (siehe 1BFAH)

; Token oder Zeichen abspeichern

1C3D EB            EX     DE,HL       ;HL = Textzeiger
1C3E 79            LD     A,C         ;A = Token
1C3F C1            POP    BC          ;Zeilenzähler zurück
1C40 D1            POP    DE          ;Bufferzeiger zurück
1C41 EB            EX     DE,HL       ;DE = Textzeiger
                                      ;HL = Bufferzeiger
1C42 FE95          CP     95H         ;ELSE-Token ?
1C44 363A          LD     (HL),3AH    ;':' in Buffer setzen
1C46 2002          JR     NZ,1C4AH    ;Nein: ':' nicht übernehmen
1C48 0C            INC    C           ;Zähler +1
1C49 23            INC    HL          ;Bufferzeiger +1
                                      ;-> ':' übernommen
1C4A FEFB          CP     0FBH        ;Apostroph ? (REM)
1C4C 200C          JR     NZ,1C5AH    ;Nein: Token übernehmen
1C4E 363A          LD     (HL),3AH    ;Ja: ':' in Buffer setzen
1C50 23            INC    HL          ;Bufferzeiger +1
1C51 0693          LD     B,93H       ;B = REM-Token
1C53 70            LD     (HL),B      ;'REM'-Token in den Buffer setzen
1C54 23            INC    HL          ;Bufferzeiger +1
1C55 EB            EX     DE,HL       ;DE = Bufferzeiger
1C56 0C            INC    C           ;Zähler +1 für ':'
1C57 0C            INC    C           ;Zähler +1 für 'REM'-Token
1C58 181D          JR     1C77H       ;Alle Zeichen bis zum Zeilenende
                                      ;übernehmen (REM-Zeile)
```

```
: Token bzw. Zeichen in A übernehmen

  1C5A EB          EX    DE,HL        ;DE = Bufferzeiger
  1C5B 23          INC   HL           ;Textzeiger +1
  1C5C 12          LD    (DE),A       ;Token im Buffer ablegen
  1C5D 13          INC   DE           ;Bufferzeiger +1
  1C5E 0C          INC   C            ;Zähler +1
  1C5F D63A        SUB   3AH          ;':' übernommen ? (SUB !!)
  1C61 2804        JR    Z,1C67H      ;Ja: Codierung freigeben und
                                      ;nächstes Zeichen codieren
  1C63 FE4E        CP    4EH          ;DATA-Token übernommen ?
                                      ;(4EH + 3AH = 88H)
  1C65 2003        JR    NZ,1C6AH     ;Nein: Flag belassen
  1C67 32B040      LD    (40B0H),A    ;Ja: Codierung sperren (A = 4EH)
  1C6A D659        SUB   59H          ;REM-Token übernommen ? (SUB !!)
  1C6C C2CC1B      JP    NZ,1BCCH     ;Nein: nächstes Zeichen codieren
  1C6F 47          LD    B,A          ;Ja: B = 0 setzen als Vergleichs-
                                      ;zeichen

: Alle Zeichen bis zum Ende der Zeile oder bis Zeichen = B ist, übernehmen

  1C70 7E          LD    A,(HL)       ;A = Textzeichen
  1C71 B7          OR    A            ;Zeilenende erreicht
  1C72 2809        JR    Z,1C7DH      ;Ja: Zeile fertig
  1C74 B8          CP    B            ;Vergleichszeichen erreicht ?
  1C75 28E4        JR    Z,1C5BH      ;Ja: Zeichen übernehmen
  1C77 23          INC   HL           ;Textzeiger +1
  1C78 12          LD    (DE),A       ;Zeichen im Buffer ablegen
  1C79 0C          INC   C            ;Zähler +1
  1C7A 13          INC   DE           ;Bufferzeiger +1
  1C7B 18F3        JR    1C70H        ;Nächstes Zeichen

: Zeile komplett codiert

  1C7D 210500      LD    HL,0005H     ;5 zur Länge hinzuaddieren
  1C80 44          LD    B,H          ;(2 Bytes Zeilenpointer, 2 Bytes
  1C81 09          ADD   HL,BC        ;Zeilennummer, 1 Byte Zeilenende)
  1C82 44          LD    B,H          ;BC = Anzahl der im Programm-
  1C83 4D          LD    C,L          ;speicher benötigten Bytes
  1C84 2AA740      LD    HL,(40A7H)   ;HL = Anfang des Zeilenbuffers
  1C87 2B          DEC   HL           ;HL = Zeiger auf codierte
  1C88 2B          DEC   HL           ;Zeile - 1
  1C89 2B          DEC   HL
  1C8A 12          LD    (DE),A       ;Codierte Zeile mit dreimal 00H
  1C8B 13          INC   DE           ;abschließen
  1C8C 12          LD    (DE),A
  1C8D 13          INC   DE
  1C8E 12          LD    (DE),A
  1C8F C9          RET
```

```
; UPRO RST 18H: 16-Bit Vergleich
; Vergleicht DE mit HL und setzt die Flags entsprechend
; (wie CP HL,DE)

  1C90 7C            LD      A,H          ;A = MSB von HL
  1C91 92            SUB     D            ;MSB von DE abziehen
  1C92 C0            RET     NZ           ;RET wenn ungleich
  1C93 7D            LD      A,L          ;Desgl. mit LSBs
  1C94 93            SUB     E
  1C95 C9            RET

; UPRO RST 08H: Syntaxprüfung
; Vergleicht das Byte in (HL) mit dem Byte das nach dem RST 08H im Speicher
; steht. Bei Gleichheit wird ein RST 10H ausgeführt und der Rücksprung erfolgt
; zur normalen RET-Adr + 1 (da bei (RET-Adr) das Vergleichsbyte steht)
; Sind die beiden Zeichen ungleich wird ein SN-Error generiert

  1C96 7E            LD      A,(HL)       ;A = Textzeichen
  1C97 E3            EX      (SP),HL      ;HL : Vergleichszeichen
  1C98 BE            CP      (HL)         ;Sind beide Zeichen gleich ?
  1C99 23            INC     HL           ;HL +1 für RET-Adr
  1C9A E3            EX      (SP),HL      ;RET-Adr zurück ins Stack
  1C9B CA781D        JP      Z,1D78H      ;RST 10H bei Gleichheit
  1C9E C39719        JP      1997H        ;sonst SN-Error

; FOR

  1CA1 3E64          LD      A,64H        ;A <> 0
  1CA3 32DC40        LD      (40DCH),A    ;Feldvariablen sperren
  1CA6 CD211F        CALL    1F21H        ;LET ausführen (Schleifenvariable
                                          ;anlegen und den Startwert
                                          ;zuweisen)
  1CA9 E3            EX      (SP),HL      ;PTZ retten, HL = RET-Adr
                                          ;DE = VARPTR auf Schleifen-
                                          ;variable
  1CAA CD3619        CALL    1936H        ;Ist bereits eine FOR-TO-Schleife
                                          ;mit dieser Variablen aktiv ?
  1CAD D1            POP     DE           ;DE = PTZ
  1CAE 2005          JR      NZ,1CB5H     ;Nein: weiter bei 1CB5H
  1CB0 09            ADD     HL,BC        ;Ja: HL = HL + 14 (BC = 000EH)
  1CB1 F9            LD      SP,HL        ;Alte Schleife mit gleicher
                                          ;Variablen aufheben (vom Stack
                                          ;löschen)
  1CB2 22E840        LD      (40E8H),HL   ;Neuen SP abspeichern
  1CB5 EB            EX      DE,HL        ;HL = PTZ
  1CB6 0E08          LD      C,08H        ;Sind noch 2*C = 16 Bytes frei ?
  1CB8 CD6319        CALL    1963H        ;(Es werden 17 Bytes gebraucht)
  1CBB E5            PUSH    HL           ;PTZ retten
  1CBC CD051F        CALL    1F05H        ;PTZ bis zum nächsten Befehl
                                          ;erhöhen (PTZ zeigt damit auf
                                          ;ersten Befehl IN der Schleife)
  1CBF E3            EX      (SP),HL      ;neuen PTZ retten, HL = alter PTZ
  1CC0 E5            PUSH    HL           ;PTZ retten
```

```
1CC1 2AA240      LD      HL,(40A2H)   ;HL = aktuelle ZN
1CC4 E3          EX      (SP),HL      ;ZN retten, PTZ zurück
1CC5 CF          RST     08H          ;'TO'-Token suchen
1CC6 BD          DEFB    'TO'-Token
1CC7 E7          RST     20H          ;TSTTYP (Startwert)
1CC8 CAF60A      JP      Z,0AF6H      ;TM-Error bei STR
1CCB D2F60A      JP      NC,0AF6H     ;TM-Error bei DBL
1CCE F5          PUSH    AF           ;Typcode - 3 retten
1CCF CD3723      CALL    2337H        ;X = Endwert
1CD2 F1          POP     AF           ;Typcode zurück
1CD3 E5          PUSH    HL           ;PTZ retten
1CD4 F2EC1C      JP      P,1CECH      ;Sprung wenn Schleifenvariable
                                      ;im SNG-Format

; INT-Schleife

1CD7 CD7F0A      CALL    0A7FH        ;Endwert ins INT-Format umwandeln
1CDA E3          EX      (SP),HL      ;Endwert retten, PTZ zurück
1CDB 110100      LD      DE,0001H     ;DE = Default-Stepwert (1)
1CDE 7E          LD      A,(HL)       ;A = Nächstes Zeichen
1CDF FECC        CP      0CCH         ;'STEP'-Token ?
1CE1 CC012B      CALL    Z,2B01H      ;Ja: DE = Stepwert
1CE4 D5          PUSH    DE           ;Stepwert retten
1CE5 E5          PUSH    HL           ;PTZ retten
1CE6 EB          EX      DE,HL        ;HL = Stepwert
1CE7 CD9E09      CALL    099EH        ;A = SGN(Stepwert)
1CEA 1822        JR      1D0EH        ;weiter bei 1D0EH

; SNG-Schleife

1CEC CDB10A      CALL    0AB1H        ;Endwert ins SNG-Format umwandeln
1CEF CDBF09      CALL    09BFH        ;BCDE = Endwert
1CF2 E1          POP     HL           ;PTZ zurück
1CF3 C5          PUSH    BC           ;Endwert retten
1CF4 D5          PUSH    DE
1CF5 010081      LD      BC,8100H     ;BCDE = Default-Stepwert (1)
1CF8 51          LD      D,C
1CF9 5A          LD      E,D
1CFA 7E          LD      A,(HL)       ;'STEP'-Token angegeben ?
1CFB FECC        CP      0CCH
1CFD 3E01        LD      A,01H        ;A = SGN(Default-Stepwert)
1CFF 200E        JR      NZ,1D0FH     ;Nein: Weiter bei 1D0FH
1D01 CD3823      CALL    2338H        ;Ja: X = Stepwert
1D04 E5          PUSH    HL           ;PTZ retten
1D05 CDB10A      CALL    0AB1H        ;X = CSNG(X)
1D08 CDBF09      CALL    09BFH        ;BCDE = X = Stepwert
1D0B CD5509      CALL    0955H        ;A = SGN(Stepwert)
```

```
; FOR-Stack abschließen

  1D0E E1              POP    HL          ;PTZ zurück
  1D0F C5              PUSH   BC          ;Stepwert retten
  1D10 D5              PUSH   DE          ;(für SNG-Schleife)
  1D11 4F              LD     C,A         ;C = SGN(Stepwert)
  1D12 E7              RST    20H         ;TSTTYP (Stepwert)
  1D13 47              LD     B,A         ;B = Typcode - 3
  1D14 C5              PUSH   BC          ;Typcode und SGN(Stepwert) retten
  1D15 E5              PUSH   HL          ;PTZ retten
  1D16 2ADF40          LD     HL,(40DFH)  ;HL = VARPTR der Schleifen-
                                          ;variablen
  1D19 E3              EX     (SP),HL     ;VARPTR retten, PTZ zurück
  1D1A 0681            LD     B,81H       ;B = 'FOR'-Token
  1D1C C5              PUSH   BC          ;Stack markieren
  1D1D 33              INC    SP  -       ;LSB wieder entfernen
; Programmschleife
; Rücksprungadresse nach Ausführung eines Befehls im Programm
  HL (PTZ) muß auf Befehlsende (':') oder Zeilenende (00H) zeigen

  1D1E CD5803          CALL   0358H       ;Taste gedrückt ?
  1D21 B7              OR     A
  1D22 C4A01D          CALL   NZ,1DA0H    ;Ja: Shift-§ oder Break ?
  1D25 22E640          LD     (40E6H),HL  ;PTZ abspeichern
  1D28 ED73E840        LD     (40E8H),SP  ;SP abspeichern
  1D2C 7E              LD     A,(HL)      ;A = nächstes Zeichen
  1D2D FE3A            CP     3AH         ;= ':' ?
  1D2F 2829            JR     Z,1D5AH     ;Ja: OK, weiter bei 1D5AH
  1D31 B7              OR     A           ;= 00H ? (Zeilenende)
  1D32 C29719          JP     NZ,1997H    ;Nein: SN-Error

; Neue Zeile beginnen

  1D35 23              INC    HL          ;PTZ +1
  1D36 7E              LD     A,(HL)      ;Adresse der nächsten Zeile
  1D37 23              INC    HL          ;prüfen
  1D38 B6              OR     (HL)        ;= 0000H (Programmende erreicht)?
  1D39 CA7E19          JP     Z,197EH     ;Ja: Programm beenden
  1D3C 23              INC    HL          ;DE = Zeilennummer
  1D3D 5E              LD     E,(HL)
  1D3E 23              INC    HL
  1D3F 56              LD     D,(HL)
  1D40 EB              EX     DE,HL       ;DE = PTZ, HL = ZN
  1D41 22A240          LD     (40A2H),HL  ;Aktuelle ZN abspeichern
  1D44 3A1B41          LD     A,(411BH)   ;TRACE aktiv ?
  1D47 B7              OR     A           ;(411BH) <> 0 ?
  1D48 280F            JR     Z,1D59H     ;Nein: weiter bei 1D59H
```

```
: TRACE ausführen

  1D4A D5              PUSH    DE          ;PTZ retten
  1D4B 3E3C            LD      A,3CH       ;A = '<'
  1D4D CD2A03          CALL    032AH       ;Zeichen ausgeben
  1D50 CDAF0F          CALL    0FAFH       ;HL als Dezimalzahl ausgeben
                                           ;(Zeilennummer)
  1D53 3E3E            LD      A,3EH       ;A = '>'
  1D55 CD2A03          CALL    032AH       ;Zeichen ausgeben
  1D58 D1              POP     DE          ;PTZ zurück

  1D59 EB              EX      DE,HL       ;HL = PTZ, DE = ZN
  1D5A D7              RST     10H         ;nächstes Zeichen nach A
  1D5B 111E1D          LD      DE,1D1EH    ;RET-Adr auf 1D1EH setzen
  1D5E D5              PUSH    DE
  1D5F C8              RET     Z           ;RET wenn Zeilenende erreicht
  1D60 D680            SUB     80H         ;Token gefunden ?
  1D62 DA211F          JP      C,1F21H     ;Nein: Zeichen als Variable auf-
                                           ;fassen, weiter bei LET
  1D65 FE3C            CP      3CH         ;Befehl oder Funktion ?
  1D67 C3C039          JP      39C0H       ;Colour-Befehl gefunden ?
  1D6A 07              RLCA                ;A = Tokenzahl * 2
  1D6B 4F              LD      C,A         ;BC = Offset für Adressentabelle
  1D6C 0600            LD      B,00H
  1D6E EB              EX      DE,HL       ;DE = PTZ
  1D6F 212218          LD      HL,1822H    ;HL = Zeiger auf Adressentabelle
  1D72 09              ADD     HL,BC       ;Offset addieren
  1D73 4E              LD      C,(HL)      ;Befehlsadresse nach BC laden
  1D74 23              INC     HL
  1D75 46              LD      B,(HL)
  1D76 C5              PUSH    BC          ;Adresse ins Stack (für RET)
  1D77 EB              EX      DE,HL       ;HL = PTZ

: RST 10H: PTZ auf nächstes Zeichen <> 20H (Leerzeichen) erhöhen
: I: HL = PTZ
: O: HL = PTZ (+1 mindestens)
:     A  = Zeichen in (HL)
:     CY = 1 wenn Ziffer gefunden
:     Z  = 1 wenn Zeilenende oder Befehlsende (':') gefunden

  1D78 23              INC     HL          ;PTZ +1
  1D79 7E              LD      A,(HL)      ;A = nächstes Zeichen
  1D7A FE3A            CP      3AH         ;Zeichen > Ziffer ?
  1D7C D0              RET     NC          ;Ja: Fertig (Z=1 wenn A = ':')
  1D7D FE20            CP      20H         ;Leerzeichen gefunden ?
  1D7F CA781D          JP      Z,1D78H     ;Ja: nächstes Zeichen holen
  1D82 FE0B            CP      0BH         ;Zeichen > 0AH ?
  1D84 3005            JR      NC,1D8BH    ;Ja: weiter bei 1D8BH
  1D86 FE09            CP      09H         ;Zeichen > 08H ?
  1D88 D2781D          JP      NC,1D78H    ;Ja: nächstes Zeichen holen
  1D8B FE30            CP      30H         ;Ziffer gefunden ?
  1D8D 3F              CCF                 ;Ja: CY = 1
  1D8E 3C              INC     A           ;Zeilenende erreicht ?
  1D8F 3D              DEC     A           ;(A = 00H ?)
                                           ;Ja: Z = 1
  1D90 C9              RET
```

```
; RESTORE

  1D91 EB             EX      DE,HL         ;DE = PTZ
  1D92 2AA440         LD      HL,(40A4H)    ;HL = Programmstart
  1D95 2B             DEC     HL            ;HL = HL - 1
  1D96 22FF40         LD      (40FFH),HL    ;DATA-Zeiger = Programmstart -1
  1D99 EB             EX      DE,HL         ;HL = PTZ
  1D9A C9             RET

; Shift-§ oder Break gedrückt ?

  1D9B CD5803         CALL    0358H         ;Taste gedrückt ?
  1D9E B7             OR      A
  1D9F C8             RET     Z             ;Nein: Fertig
  1DA0 FE60           CP      60H           ;Ja: Shift-§ gedrückt ?
  1DA2 CC8403         CALL    Z,0384H       ;Ja: auf nächsten Tastendruck
                                            ;warten
  1DA5 329940         LD      (4099H),A     ;Letzten Tastencode abspeichern
  1DA8 3D             DEC     A             ;Break ? (Tastencode = 01H)

; STOP

  1DA9 C0             RET     NZ            ;Nein: Fertig / SN-Error ?

; Break gedrückt

  1DAA 3C             INC     A             ;A = 01H (Tastencode von Break)
  1DAB C3B41D         JP      1DB4H         ;weiter bei 1DB4H

; END

  1DAE C0             RET     NZ            ;SN-Error ?
  1DAF F5             PUSH    AF            ;A retten (A ist 00H !)
  1DB0 CCBB41         CALL    Z,41BBH       ;DOS
  1DB3 F1             POP     AF            ;A zurück

; STOP (Z=0) , END (Z=1)

  1DB4 22E640         LD      (40E6H),HL    ;PTZ abspeichern
  1DB7 21B540         LD      HL,40B5H      ;Stringtabellenzeiger
  1DBA 22B340         LD      (40B3H),HL    ;zurücksetzen
  1DBD 21F6FF         LD      HL,0FFF6H     ;--
```

```
: STOP-Einsprung wenn bei INPUT Break gedrückt wurde

*1DBE    F6FF       OR      0FFH          ;A <> 0, Z = 0
 1DC0 C1            POP     BC            ;RET-Adr löschen
 1DC1 2AA240        LD      HL,(40A2H)    ;HL = Aktuelle ZN
 1DC4 E5            PUSH    HL            ;ZN retten
 1DC5 F5            PUSH    AF            ;Flags retten
 1DC6 7D            LD      A,L           ;ZN = 65535 ?
 1DC7 A4            AND     H             ;(aktiver Befehlsmodus)
 1DC8 3C            INC     A
 1DC9 2809          JR      Z,1DD4H       ;Ja: kein CONT möglich
 1DCB 22F540        LD      (40F5H),HL    ;Sonst Zeilennummer
 1DCE 2AE640        LD      HL,(40E6H)    ;und PTZ für CONT abspeichern
 1DD1 22F740        LD      (40F7H),HL
 1DD4 CD8B03        CALL    038BH         ;Druckerausgabe beenden
 1DD7 CDF920        CALL    20F9H         ;Neue Zeile beginnen
 1DDA F1            POP     AF            ;Flags zurück
 1DDB 213019        LD      HL,1930H      ;HL = Zeiger auf Text 'Break '
 1DDE C2061A        JP      NZ,1A06H      ;STOP: weiter bei 1A06H
 1DE1 C3181A        JP      1A18H         ;END: Zurück zum aktiven
                                          ;Befehlsmodus

: CONT

 1DE4 2AF740        LD      HL,(40F7H)    ;HL = alter PTZ
 1DE7 7C            LD      A,H           ;HL = 0000H ?
 1DE8 B5            OR      L
 1DE9 1E20          LD      E,20H         ;E = Fehlercode für CN-Error
 1DEB CAA219        JP      Z,19A2H       ;Ja: CN-Error
 1DEE EB            EX      DE,HL         ;DE = PTZ
 1DEF 2AF540        LD      HL,(40F5H)    ;HL = alte Zeilennummer
 1DF2 CDA038        CALL    38A0H         ;HL als aktuelle ZN abspeichern
                                          ;und CRTC auf letzten Wert pro-
                                          ;grammieren
 1DF5 EB            EX      DE,HL         ;HL = PTZ
 1DF6 C9            RET                   ;nächsten Befehl ausführen

: TRON

 1DF7 3EAF          LD      A,0AFH        ;A <> 0

: TROFF

*1DF8    AF         XOR     A             ;A = 0
 1DF9 321B41        LD      (411BH),A     ;TRACE-Flag setzen
 1DFC C9            RET                   ;Fertig

 1DFD F1            POP     AF            ;--
 1DFE E1            POP     HL
 1DFF C9            RET
```

```
: DEFSTR

 1E00 1E03          LD      E,03H       ;E = VT für STR
 1E02 011E02        LD      BC,021EH    ;--

: DEFINT

*1E03   1E02        LD      E,02H       ;E = VT für INT
 1E05 011E04        LD      BC,041EH    ;--

: DEFSNG

*1E06   1E04        LD      E,04H       ;E = VT für SNG
 1E08 011E08        LD      BC,081EH    ;--

: DEFDBL

*1E09   1E08        LD      E,08H       ;E = VT für DBL
 1E0B CD3D1E        CALL    1E3DH       ;Buchstabe angegeben ?
 1E0E 019719        LD      BC,1997H    ;BC = RET-Adr auf SN-Error
 1E11 C5            PUSH    BC          ;BC als RET-Adr setzen
 1E12 D8            RET     C           ;SN-Error wenn kein Buchstabe
                                        ;angegeben
 1E13 D641          SUB     41H         ;A = Offset für Tabelle
 1E15 4F            LD      C,A         ;C = A
 1E16 47            LD      B,A         ;B = A
 1E17 D7            RST     10H         ;nächstes Zeichen holen
 1E18 FECE          CP      0CEH        ;'-' gefunden ?
 1E1A 2009          JR      NZ,1E25H    ;Nein: Tabelle setzen
 1E1C D7            RST     10H         ;Ja: 2. Buchstaben holen
 1E1D CD3D1E        CALL    1E3DH       ;Buchstabe gefunden ?
 1E20 D8            RET     C           ;Nein: SN-Error
 1E21 D641          SUB     41H         ;A = Offset für Tabelle
 1E23 47            LD      B,A         ;B = A
 1E24 D7            RST     10H         ;PTZ auf nächstes Zeichen erhöhen
 1E25 78            LD      A,B         ;A = Offset des 2.Buchstaben
 1E26 91            SUB     C           ; - Offset des 1.Buchstaben
 1E27 D8            RET     C           ;SN-Error wenn die Buchstaben
                                        ;nicht in alphabetischer Reihen-
                                        ;folge angegeben wurden
 1E28 3C            INC     A           ;A = Zähler
 1E29 E3            EX      (SP),HL     ;PTZ retten, RET-Adr löschen
 1E2A 210141        LD      HL,4101H    ;HL = Tabellenzeiger
 1E2D 0600          LD      B,00H       ;BC = Offset auf 1.Buchstaben
 1E2F 09            ADD     HL,BC       ;HL = Zeiger auf 1.Buchstaben
 1E30 73            LD      (HL),E      ;VT in Tabelle setzen
 1E31 23            INC     HL          ;Zeiger +1
 1E32 3D            DEC     A           ;Zähler -1
 1E33 20FB          JR      NZ,1E30H    ;nächste Tabellenposition setzen
 1E35 E1            POP     HL          ;PTZ zurück
 1E36 7E            LD      A,(HL)      ;A = nächstes Zeichen
 1E37 FE2C          CP      2CH         ;',' angegeben ?
 1E39 C0            RET     NZ          ;Nein: Fertig
 1E3A D7            RST     10H         ;Ja: Nächstes Zeichen holen
 1E3B 18CE          JR      1E0BH       ;und DEF wiederholen
```

```
: Test ob das ASCII-Zeichen in (HL) ein Grossbuchstabe ist
: I: HL : zu testendes ASCII-Zeichen
: O: CY = 0 wenn das Zeichen ein Grossbuchstabe ist (sonst CY = 1)

  1E3D 7E              LD     A,(HL)       ;A = Zeichen
  1E3E FE41            CP     41H          ;A < 'A' ?
  1E40 D8              RET    C            ;Ja: RET mit CY = 1
  1E41 FE5B            CP     5BH          ;A > 'Z' ?
  1E43 3F              CCF                 ;Ja: CY = 1
  1E44 C9              RET                 ;Fertig

; Argument in (HL) in INT-Zahl umwandeln. FC-Error bei negativer Zahl

  1E45 D7              RST    10H          ;PTZ erhöhen
  1E46 CD022B          CALL   2B02H        ;Argument holen
  1E49 F0              RET    P            ;RET wenn Positiv

; FC-Error

  1E4A 1E08            LD     E,08H        ;E = Fehlercode
  1E4C C3A219          JP     19A2H        ;Fehlerroutine anspringen

: Zeilennummer decodieren (Zahl oder '.')

  1E4F 7E              LD     A,(HL)       ;A = Zeichen
  1E50 FE2E            CP     2EH          ;'.' gefunden ?
  1E52 EB              EX     DE,HL
  1E53 2AEC40          LD     HL,(40ECH)   ;DE = '.'-Zeilennummer
  1E56 EB              EX     DE,HL
  1E57 CA781D          JP     Z,1D78H      ;Ja: RST 10H ausführen. Fertig

: Nummer in (HL) nach DE decodieren

  1E5A 2B              DEC    HL           ;PTZ -1 (wegen RST 10H)
  1E5B 110000          LD     DE,0000H     ;Ergebnis = 0
  1E5E D7              RST    10H          ;nächste Zeichen holen
  1E5F D0              RET    NC           ;Fertig wenn keine Ziffer
                                           ;gefunden
  1E60 E5              PUSH   HL           ;PTZ retten
  1E61 F5              PUSH   AF           ;Ziffer retten
  1E62 219819          LD     HL,1998H     ;HL = 6552
  1E65 DF              RST    18H          ;Ist das Ergebnis schon jetzt
                                           ;größer als 6552 ?
  1E66 DA9719          JP     C,1997H      ;Ja: SN-Error (mit der jetzigen
                                           ;Ziffer wäre DE größer als 65529)
```

```
1E69 62                  LD      H,D           ;HL = alter Wert
1E6A 6B                  LD      L,E
1E6B 19                  ADD     HL,DE         ;HL = HL * 10
1E6C 29                  ADD     HL,HL
1E6D 19                  ADD     HL,DE
1E6E 29                  ADD     HL,HL
1E6F F1                  POP     AF            ;Ziffer zurück
1E70 D630                SUB     30H           ;A = Ziffernwert
1E72 5F                  LD      E,A           ;DE = Ziffernwert
1E73 1600                LD      D,00H
1E75 19                  ADD     HL,DE         ;Neuen Ziffernwert aufaddieren
1E76 EB                  EX      DE,HL         ;DE = neuer Wert
1E77 E1                  POP     HL            ;PTZ zurück
1E78 18E4                JR      1E5EH         ;nächste Ziffer holen

; CLEAR

1E7A CA611B              JP      Z,1B61H       ;nach 1B61H wenn kein Argument
                                               ;angegeben wurde

; CLEAR mit Argument

1E7D CD461E              CALL    1E46H         ;INT-Arg holen, FC-Error wenn < 0
1E80 2B                  DEC     HL            ;PTZ -1
1E81 D7                  RST     10H           ;Befehlsende erreicht ?
1E82 C0                  RET     NZ            ;Nein: SN-Error
1E83 E5                  PUSH    HL            ;PTZ retten
1E84 2AB140              LD      HL,(40B1H)    ;HL : Höchster Speicherplatz
1E87 7D                  LD      A,L           ;DE : Höchster Speicherplatz -
                                               ;Größe des neuen Stringspeichers
1E88 93                  SUB     E
1E89 5F                  LD      E,A
1E8A 7C                  LD      A,H
1E8B 9A                  SBC     A,D
1E8C 57                  LD      D,A
1E8D DA7A19              JP      C,197AH       ;OM-Error wenn Stringspeicher
                                               ;zu groß
1E90 2AF940              LD      HL,(40F9H)    ;HL = Zeiger auf Programmende
1E93 012800              LD      BC,0028H      ;BC = 40
1E96 09                  ADD     HL,BC         ;HL : Programmende + 40
1E97 DF                  RST     18H           ;HL und DE vergleichen
1E98 D27A19              JP      NC,197AH      ;OM-Error wenn weniger als
                                               ;40 Bytes Platz
1E9B EB                  EX      DE,HL         ;HL : Neuer Anfang des String-
                                               ;speichers
1E9C 22A040              LD      (40A0H),HL    ;HL abspeichern
1E9F E1                  POP     HL            ;PTZ zurück
1EA0 C3611B              JP      1B61H         ;CLEAR ausführen, Fertig
```

```
: RUN

  1EA3 CA5D1B          JP     Z.1B5DH       :weiter bei 1B5DH wenn keine
                                            :Zeilennummer angegeben wurde
  1EA6 CDC741          CALL   41C7H         :DOS
  1EA9 CD611B          CALL   1B61H         :CLEAR
  1EAC 011E1D          LD     BC.1D1EH      :BC = RET-Adr auf Programm-
                                            :schleife
  1EAF 1810            JR     1EC1H         :weiter bei GOTO

: GOSUB

  1EB1 0E03            LD     C.03H         :Noch 2 x C Bytes frei ?
  1EB3 CD6319          CALL   1963H         :Sonst OM-Error
  1EB6 C1              POP    BC            :BC = RET-Adr
  1EB7 E5              PUSH   HL            :PTZ retten (für GOSUB-Stack)
  1EB8 E5              PUSH   HL            :PTZ retten
  1EB9 2AA240          LD     HL.(40A2H)    :HL = Aktuelle Zeilennummer
  1EBC E3              EX     (SP).HL       :Aktuelle ZN retten. PTZ zurück
  1EBD 3E91            LD     A.91H         :A = GOSUB-Token
  1EBF F5              PUSH   AF            :A als Kennung in Stack
  1EC0 33              INC    SP            :LSB aus Stack löschen
  1EC1 C5              PUSH   BC            :RET-Adr zurück in Stack

: GOTO

  1EC2 CD5A1E          CALL   1E5AH         :ZN decodieren. DE = ZN
  1EC5 CD071F          CALL   1F07H         :PTZ bis zum Ende der Zeile
                                            :erhöhen
  1EC8 E5              PUSH   HL            :neuen PTZ retten
  1EC9 2AA240          LD     HL.(40A2H)    :HL = Aktuelle ZN
  1ECC DF              RST    18H           :Aktuelle ZN mit gesuchter ZN
                                            :vergleichen
  1ECD E1              POP    HL            :neuen PTZ zurück
  1ECE 23              INC    HL            :HL = Zeiger auf nächste Zeile
  1ECF DC2F1B          CALL   C.1B2FH       :Zeile DE ab HL suchen. wenn
                                            :gesuchte ZN > aktuelle ZN
  1ED2 D42C1B          CALL   NC.1B2CH      :Zeile DE ab Programmstart
                                            :suchen. wenn gesuchte ZN <=
                                            :aktuelle ZN
  1ED5 60              LD     H.B           :HL = Zeiger auf gesuchte Zeile
  1ED6 69              LD     L.C
  1ED7 2B              DEC    HL            :HL -1
  1ED8 D8              RET    C             :Fertig wenn Zeile gefunden

: UL-Error

  1ED9 1E0E            LD     E.0EH         :E = Fehlercode
  1EDB C3A219          JP     19A2H         :Fehlerroutine anspringen
```

```
: RETURN

  1EDE C0              RET    NZ             ;SN-Error ?
  1EDF 16FF            LD     D,0FFH         ;Flag <> 0
  1EE1 CD3619          CALL   1936H          ;GOSUB-Stack suchen
  1EE4 F9              LD     SP,HL          ;SP = HL (GOSUB-Stack löschen)
  1EE5 22E840          LD     (40E8H),HL     ;Neuen SP abspeichern
  1EE8 FE91            CP     91H            ;GSOUB-Stack gefunden ?
  1EEA 1E04            LD     E,04H          ;E = Fehlercode für RG-Error
  1EEC C2A219          JP     NZ,19A2H       ;Nein: RG-Error
  1EEF E1              POP    HL             ;HL = ZN der GOSUB-Zeile
  1EF0 22A240          LD     (40A2H),HL     ;Als aktuelle ZN setzen
  1EF3 23              INC    HL             ;ZN = 65535 ?
  1EF4 7C              LD     A,H            ;(kam der GOSUB-Befehl vom
  1EF5 B5              OR     L              ;aktiven Befehlsmodus ?)
  1EF6 2007            JR     NZ,1EFFH       ;Nein: PTZ auf nächsten Befehl
                                             ;nach 'GOSUB' erhöhen und dort
                                             ;Programmausführung fortsetzen
  1EF8 3ADD40          LD     A,(40DDH)      ;STOP-Flag = 0 ?
  1EFB B7              OR     A
  1EFC C2181A          JP     NZ,1A18H       ;Nein: Zurück zum aktiven
                                             ;Befehlsmodus
  1EFF 211E1D          LD     HL,1D1EH       ;HL = RET-Adr zur Programm-
                                             ;schleife
  1F02 E3              EX     (SP),HL        ;HL = PTZ, RET-Adr= 1D1EH
  1F03 3EE1            LD     A,0E1H         ;--
 *1F04   E1            POP    HL             ;PTZ zurück

: PTZ (HL) auf nächsten Befehl erhöhen

  1F05 013A0E          LD     BC,0E3AH       ;C = 3AH (':')

: PTZ (HL) auf nächste Zeile erhöhen

**1F07     0E00        LD     C,00H          ;C = 00H
 *1F08 00              NOP                   ;--
  1F09 0600            LD     B,00H          ;B = 00H
  1F0B 79              LD     A,C            ;A = Suchzeichen (= 00H wenn
                                             ;String gefunden, PTZ wird dann
                                             ;aufs Ende der Zeile erhöht)
  1F0C 48              LD     C,B            ;C = 00H (= Suchzeichen wenn
                                             ;String gefunden)
  1F0D 47              LD     B,A            ;B = Suchzeichen
  1F0E 7E              LD     A,(HL)         ;A = nächstes Zeichen
  1F0F B7              OR     A              ;Zeilenende ?
  1F10 C8              RET    Z              ;Ja: Fertig
  1F11 B8              CP     B              ;Zeichen gefunden ?
  1F12 C8              RET    Z              ;Ja: Fertig
  1F13 23              INC    HL             ;PTZ +1
  1F14 FE22            CP     22H            ;String gefunden ?
  1F16 28F3            JR     Z,1F0BH        ;Ja: PTZ bis Zeilenende erhöhen
```

```
1F18 D68F           SUB    8FH           ;'IF' gefunden ?
1F1A 20F2           JR     NZ,1F0EH      ;Nein: weitersuchen
1F1C B8             CP     B             ;Ja: CY = 1 wenn 3AH gesucht
                                         ;wurde (A = 00H !)
1F1D 8A             ADC    A,D           ;A = 00H + D + CY
1F1E 57             LD     D,A           ;D = Zähler für geschachtelte
                                         ;IF THEN ELSE-Befehle
1F1F 18ED           JR     1F0EH         ;weitersuchen

; LET

1F21 CD0D26         CALL   260DH         ;DE = VARPTR der Variablen
1F24 CF             RST    08H           ;Es muß '=' folgen
1F25 D5             DEFB   '='-Token
1F26 EB             EX     DE,HL
1F27 22DF40         LD     (40DFH),HL    ;VARPTR abspeichern
1F2A EB             EX     DE,HL
1F2B D5             PUSH   DE            ;VARPTR retten
1F2C E7             RST    20H           ;TSTTYP
1F2D F5             PUSH   AF            ;Typcode-3 retten
1F2E CD3723         CALL   2337H         ;Ausdruck nach X
1F31 F1             POP    AF            ;Typcode-3 zurück
1F32 E3             EX     (SP),HL       ;PTZ retten, VARPTR zurück
1F33 C603           ADD    A,03H         ;Typcode berichtigen
1F35 CD1928         CALL   2819H         ;X in gewünschten Typ umwandeln
1F38 CD030A         CALL   0A03H         ;DE = Zeiger auf LSB(X) für
                                         ;SNG,DBL und INT, TSTTYP
1F3B E5             PUSH   HL            ;VARPTR retten
1F3C 2028           JR     NZ,1F66H      ;Sprung wenn nicht STR

; Wertzuweisung auf Stringvariable

1F3E 2A2141         LD     HL,(4121H)    ;HL: Vektor des neuen Strings
1F41 E5             PUSH   HL            ;HL retten
1F42 23             INC    HL
1F43 5E             LD     E,(HL)        ;DE = Zeiger auf neuen String
1F44 23             INC    HL
1F45 56             LD     D,(HL)
1F46 2AA440         LD     HL,(40A4H)    ;HL = Startadresse des Basic-
                                         ;programms
1F49 DF             RST    18H           ;CP HL,DE
1F4A 300E           JR     NC,1F5AH      ;Sprung wenn Stringadresse
                                         ; < Programmstart (String ist im
                                         ;Zeilenbuffer) (z.B. bei INPUT)
1F4C 2AA040         LD     HL,(40A0H)    ;HL = Startadresse des String-
                                         ;speichers
1F4F DF             RST    18H           ;CP HL,DE
1F50 D1             POP    DE            ;DE: Vektor des neuen Strings
1F51 300F           JR     NC,1F62H      ;Sprung wenn Stringadresse
                                         ; < Stringspeichers (String ist
                                         ;im Programmtext)
                                         ;(Stringkonstante)
```

```
1F53 2AF940     LD      HL,(40F9H)   ;HL = Startadresse der Variablen
1F56 DF         RST     18H          ;CP HL,DE
1F57 3009       JR      NC,1F62H     ;Sprung wenn Vektorenadresse des
                                     ;neuen Strings nicht im
                                     ;Variablenspeicher liegt (der
                                     ;neue String ist keine Variable)
1F59 3ED1       LD      A,0D1H       ;--
*1F5A  D1       POP     DE           ;DE: Vektor des neuen Strings
1F5B CDF529     CALL    29F5H        ;BC = Adresse des letzten Strings
                                     ;in der Stringtabelle
1F5E EB         EX      DE,HL        ;DE: Vektor des letzten Strings
                                     ;in Stringtabelle, HL: Vektor
                                     ;des neuen Strings
1F5F CD4328     CALL    2843H        ;Neuen String in Stringspeicher
                                     ;übernehmen, Stringvektor in die
                                     ;Stringtabelle eintragen
1F62 CDF529     CALL    29F5H        ;DE: Vektor des neuen Strings
1F65 E3         EX      (SP),HL      ;HL: Vektor der Variablen
                                     ;Vektor des neuen Strings retten
1F66 CDD309     CALL    09D3H        ;VT-Bytes von (DE) nach (HL)
                                     ;kopieren
1F69 D1         POP     DE           ;Vektor des neuen Strings zurück
1F6A E1         POP     HL           ;PTZ zurück
1F6B C9         RET

; ON

1F6C FE9E       CP      9EH          ;nächstes Token = 'ERROR' ?
1F6E 2025       JR      NZ,1F95H     ;Nein: weiter bei 1F95H

; ON ERROR GOTO

1F70 D7         RST     10H          ;PTZ erhöhen
1F71 CF         RST     08H          ;nächsten Token muß 'GOTO' sein
1F72 8D         DEFB    'GOTO'-Token
1F73 CD5A1E     CALL    1E5AH        ;Zeilennummer decodieren
1F76 7A         LD      A,D          ;Zeilennummer = 0 ?
1F77 B3         OR      E
1F78 2809       JR      Z,1F83H      ;Ja: weiter bei 1F83H
1F7A CD2A1B     CALL    1B2AH        ;PTZ retten und Zeile DE suchen
1F7D 50         LD      D,B          ;DE = Zeilenadresse
1F7E 59         LD      E,C
1F7F E1         POP     HL           ;PTZ zurück
1F80 D2D91E     JP      NC,1ED9H     ;UL-Error wenn Zeile nicht
                                     ;gefunden
1F83 EB         EX      DE,HL
1F84 22F040     LD      (40F0H),HL   ;Zeilennummer abspeichern
1F87 EB         EX      DE,HL
1F88 D8         RET     C            ;Fertig wenn ZN <> 0
1F89 3AF240     LD      A,(40F2H)    ;A = Errorflag
1F8C B7         OR      A            ;ON-ERROR-GOTO gesetzt ?
1F8D C8         RET     Z            ;Nein: OK
1F8E 3A9A40     LD      A,(409AH)    ;Ja: A = letzter Fehlercode
1F91 5F         LD      E,A          ;E = Fehlercode
1F92 C3AB19     JP      19ABH        ;Fehler bearbeiten
```

```
; ON GOTO / GOSUB

  1F95 CD1C2B        CALL    2B1CH        ;DE = Argument (0 - 255)
  1F98 7E            LD      A,(HL)       ;A = nächstes Zeichen
  1F99 47            LD      B,A          ;B = Zeichen
  1F9A FE91          CP      91H          ;'GOSUB'-Token ?
  1F9C 2803          JR      Z,1FA1H      ;Ja: weiter bei 1FA1H
  1F9E CF            RST     08H          ;Nein: Es muß 'GOTO' sein
  1F9F 8D            DEFB    'GOTO'-Token
  1FA0 2B            DEC     HL           ;PTZ -1 (wegen RST 08H)
  1FA1 4B            LD      C,E          ;C = Argument (Zähler)
  1FA2 0D            DEC     C            ;Zähler -1
  1FA3 78            LD      A,B          ;A = Token
  1FA4 CA601D        JP      Z,1D60H      ;Zähler = 0 ? Ja: HL zeigt auf
                                          ;gewünschte Zeilennummer, Befehl
                                 -        ;ausführen
  1FA7 CD5B1E        CALL    1E5BH        ;Durch Decodierung der Nummer in
                                          ;(HL), PTZ auf nächstes Zeichen
                                          ;nach der Nummer erhöhen
  1FAA FE2C          CP      2CH          ;Das Trennungszeichen muß ','
                                          ;sein
  1FAC C0            RET     NZ           ;Sonst nächsten Befehl ausführen
  1FAD 18F3          JR      1FA2H        ;Gewünschte Zeilennummer
                                          ;erreicht ?

; RESUME

  1FAF 11F240        LD      DE,40F2H     ;DE : ON ERROR GOTO-Flag
  1FB2 1A            LD      A,(DE)       ;A = ON ERROR GOTO-Flag
  1FB3 B7            OR      A            ;ON-ERROR-GOTO aktiv ?
  1FB4 CAA019        JP      Z,19A0H      ;Nein: RW-Error
  1FB7 3C            INC     A            ;A +1
  1FB8 329A40        LD      (409AH),A    ;letzten Fehlercode <> 0 setzen
  1FBB 12            LD      (DE),A       ;Errorflag <> 0 setzen
  1FBC 7E            LD      A,(HL)       ;A = nächstes Programmzeichen
  1FBD FE87          CP      87H          ;'NEXT'-Token ?
  1FBF 280C          JR      Z,1FCDH      ;Ja: weiter bei 1FCDH
  1FC1 CD5A1E        CALL    1E5AH        ;Zeilennummer decodieren
  1FC4 C0            RET     NZ           ;SN-Error ?
  1FC5 7A            LD      A,D          ;Zeilennummer = 0 ?
  1FC6 B3            OR      E
  1FC7 C2C51E        JP      NZ,1EC5H     ;Nein: GOTO ausführen
  1FCA 3C            INC     A            ;Ja: A = 1 (Z = 0)
  1FCB 1802          JR      1FCFH        ;RESUME NEXT ausführen
```

: RESUME NEXT

```
1FCD D7          RST    10H           ;PTZ erhöhen
1FCE C0          RET    NZ            ;SN-Error ?
1FCF 2AEE40      LD     HL,(40EEH)    ;HL = PTZ auf nächsten Befehl
1FD2 EB          EX     DE,HL         ;DE = HL
1FD3 2AEA40      LD     HL,(40EAH)    ;HL = ZN der Fehlerzeile
1FD6 22A240      LD     (40A2H),HL    ;als aktuelle ZN abspeichern
1FD9 EB          EX     DE,HL         ;HL = PTZ
1FDA C0          RET    NZ            ;Fertig bei RESUME 0
1FDB 7E          LD     A,(HL)        ;A = nächstes Zeichen im
                                      ;Programmtext
1FDC B7          OR     A             ;Zeilenende ?
1FDD 2004        JR     NZ,1FE3H      ;Nein: weiter bei 1FE3H
1FDF 23          INC    HL            ;Ja: PTZ auf nächste Zeile
1FE0 23          INC    HL   -        ;erhöhen
1FE1 23          INC    HL
1FE2 23          INC    HL
1FE3 23          INC    HL            ;PTZ +1
1FE4 7A          LD     A,D           ;Fehlerzeile = 65535 ?
1FE5 A3          AND    E
1FE6 3C          INC    A
1FE7 C2051F      JP     NZ,1F05H      ;Nein: PTZ auf nächsten Befehl
                                      ;erhöhen und Programm fortsetzen
1FEA 3ADD40      LD     A,(40DDH)     ;A = STOP-Flag
1FED 3D          DEC    A             ;A = 1 ?
1FEE CABE1D      JP     Z,1DBEH       ;Ja: STOP
1FF1 C3051F      JP     1F05H         ;Nein: Programm beim nächsten
                                      ;Befehl fortsetzen
```

: ERROR

```
1FF4 CD1C2B      CALL   2B1CH         ;DE = Argument (0-255)
1FF7 C0          RET    NZ            ;SN-Error ?
1FF8 B7          OR     A             ;Zahl = 0 ? (ERROR 0 ?)
1FF9 CA4A1E      JP     Z,1E4AH       ;Ja: FC-Error
1FFC 3D          DEC    A             ;A = A - 1
1FFD 87          ADD    A,A           ;A = A * 2
1FFE 5F          LD     E,A           ;E = (Argument - 1) * 2
1FFF FE2D        CP     2DH           ;Fehlercode > 44 ?
2001 3802        JR     C,2005H       ;Nein: Fehlercode bearbeiten
2003 1E26        LD     E,26H         ;Ja: E = Fehlercode für UE-Error
2005 C3A219      JP     19A2H         ;Zur Fehlerroutine springen
```

: AUTO

```
2008 110A00      LD     DE,000AH      ;DE = Default Start-ZN
200B D5          PUSH   DE            ;Start-ZN retten
200C 2817        JR     Z,2025H       ;weiter bei 2025H wenn keine
                                      ;Nummern angegeben wurden
```

```
200E CD4F1E         CALL   1E4FH         ;DE = Angegebene Start-ZN
2011 EB             EX     DE,HL         ;HL = ZN, DE = PTZ
2012 E3             EX     (SP),HL       ;Start-ZN retten, HL = Default-
                                         ;Start-ZN
2013 2811           JR     Z,2026H       ;Sprung wenn kein Abstand
                                         ;angegeben
2015 EB             EX     DE,HL         ;PTZ zurück nach HL
2016 CF             RST    08H           ;Jetzt muß ',' folgen
2017 2C             DEFB   ','
2018 EB             EX     DE,HL
2019 2AE440         LD     HL,(40E4H)    ;DE = Alter Abstand
201C EB             EX     DE,HL
201D 2806           JR     Z,2025H       ;Sprung wenn alter Abstand
                                         ;übernommen werden soll
                                         ;(AUTO XX,)
201F CD5A1E         CALL   1E5AH         ;DE = Abstand
2022 C29719         JP     NZ,1997H      ;SN-Error ?
2025 EB             EX     DE,HL         ;DE = PTZ, HL = Abstand
2026 7C             LD     A,H           ;Abstand = 0 ?
2027 B5             OR     L
2028 CA4A1E         JP     Z,1E4AH       ;Ja: FC-Error
202B 22E440         LD     (40E4H),HL    ;Abstand abspeichern
202E 32E140         LD     (40E1H),A     ;AUTO-Flag <> 0 setzen
2031 E1             POP    HL            ;Start-ZN zurück
2032 22E240         LD     (40E2H),HL    ;und abspeichern
2035 C1             POP    BC            ;RET-Adr löschen
2036 C3331A         JP     1A33H         ;Zeile eingeben lassen

; IF

2039 CD3723         CALL   2337H         ;X = Ausdruck
                                         ;(X = 0 wenn Bedingung nicht er-
                                         ;füllt, sonst X = -1)
203C 7E             LD     A,(HL)        ;nächstes Zeichen holen
203D FE2C           CP     2CH           ;Ist es ',' ?
203F CC781D         CALL   Z,1D78H       ;Ja: PTZ erhöhen (wie 'THEN')
2042 FECA           CP     0CAH          ;Ist es das 'THEN'-Token ?
2044 CC781D         CALL   Z,1D78H       ;Ja: PTZ erhöhen
2047 2B             DEC    HL            ;PTZ -1
2048 E5             PUSH   HL            ;PTZ retten
2049 CD9409         CALL   0994H         ;TEST1
204C E1             POP    HL            ;PTZ zurück
204D 2807           JR     Z,2056H       ;weiter bei 2056H wenn X = 0
                                         ;(Bedingung nicht erfüllt)

; IF-Bedingung erfüllt

204F D7             RST    10H           ;PTZ erhöhen
2050 DAC21E         JP     C,1EC2H       ;Sprung nach 'GOTO' wenn Zahl
                                         ;angegeben
2053 C35F1D         JP     1D5FH         ;Sonst Befehlstoken ausführen
```

```
; IF-Bedingung nicht erfüllt

  2056 1601            LD      D,01H          :D = Zähler für geschachtelte
                                               ;IF THEN ELSE-Bedingungen
  2058 CD051F          CALL    1F05H          :PTZ auf nächsten Befehl erhöhen
  205B B7              OR      A              :Zeilenende erreicht ?
  205C C8              RET     Z              ;Ja: Fertig
  205D D7              RST     10H            :Nein: Zeichen nach A holen
  205E FE95            CP      95H            :'ELSE'-Token gefunden ?
  2060 20F6            JR      NZ,2058H       :Nein: weitersuchen
  2062 15              DEC     D              :Schachtelungszähler -1
  2063 20F3            JR      NZ,2058H       ;äußerstes 'ELSE' suchen
  2065 18E8            JR      204FH          :und nachfolgenden Befehl aus-
                                               :führen

; LPRINT

  2067 3E01            LD      A,01H          :A = 01H
  2069 329C40          LD      (409CH),A      ;Nachfolgende Ausgaben auf
                                               :Drucker
  206C C39B20          JP      209BH          ;weiter bei PRINT

; PRINT

  206F CDCA41          CALL    41CAH          :DOS
  2072 FE40            CP      40H            :PRINT § ?
  2074 2019            JR      NZ,208FH       ;Nein: weiter bei 208FH

; PRINT §

  2076 CD012B          CALL    2B01H          ;Argument holen
  2079 E5              PUSH    HL             ;PTZ retten
  207A C3D430          JP      30D4H          ;Argument testen und PTZ zurück
  207D 00              NOP                    ;(Rücksprung erfolgt nach 207EH)
  207E E5              PUSH    HL             :PTZ retten
  207F 210044          LD      HL,4400H       ;HL = Startadresse des Bild-
                                               ;schirmspeichers
  2082 19              ADD     HL,DE          :HL = Startadresse + §-Argument
  2083 222040          LD      (4020H),HL     ;als neue Cursorposition
                                               ;abspeichern
  2086 CD2A36          CALL    362AH          ;neue POS errechnen
  2089 32A640          LD      (40A6H),A      ;und abspeichern
  208C E1              POP     HL             ;PTZ zurück
  208D CF              RST     08H            :Nach '§' muß ',' folgen
  208E 2C              DEFB    ','

  208F FE23            CP      23H            :PRINT # ?
  2091 2008            JR      NZ,209BH       :Nein: weiter bei 209BH
  2093 CDA935          CALL    35A9H          :Ja: Leader und Sync schreiben
  2096 3E30            LD      A,30H          :Ausgabeflag auf Cassettenausgabe
  2099 329C40          LD      (409CH),A      :setzen
```

```
209B 2B          DEC    HL            :PTZ -1
209C D7          RST    10H           :nächstes Zeichen nach A
209D CCFE20      CALL   Z.20FEH       :PRINT abschließen wenn kein
                                      :Argument gefunden
20A0 CA6921      JP     Z.2169H       :und nächste Ausgabe wieder zum
                                      :Bildschirm leiten
20A3 FEBF        CP     0BFH          :'USING'-Token ?
20A5 CABD2C      JP     Z.2CBDH       :Ja: weiter bei 2CBDH
20A8 FEBC        CP     0BCH          :'TAB('-Token ?
20AA CA3721      JP     Z.2137H       :Ja: weiter bei 2137H
20AD E5          PUSH   HL            :PTZ retten
20AE FE2C        CP     2CH           :',' gefunden ?
20B0 CA0821      JP     Z.2108H       :Ja: weiter bei 2108H
20B3 FE3B        CP     3BH           :';' gefunden ?
20B5 CA6421      JP     Z.2164H       :Ja: weiter bei 2164H
20B8 C1          POP    BC            :Stack berichtigen (PTZ zurück)
20B9 CD3723      CALL   2337H         :Argument nach X holen
20BC E5          PUSH   HL            :PTZ retten
20BD E7          RST    20H           :TSTTYP
20BE 2832        JR     Z.20F2H       :weiter bei 20F2H wenn String
20C0 CDBD0F      CALL   0FBDH         :Umwandlung Zahl -> String
20C3 CD6528      CALL   2865H         :String übernehmen
20C6 CDCD41      CALL   41CDH         :DOS
20C9 2A2141      LD     HL,(4121H)    :HL: Vektor
20CC 3A9C40      LD     A,(409CH)     :Ausgabeflag testen
20CF B7          OR     A
20D0 FAE920      JP     M.20E9H       :weiter bei 20E9H bei Cassetten-
                                      :ausgabe
20D3 2808        JR     Z.20DDH       :weiter bei 20DDH bei Bildschirm-
                                      :ausgabe

: Druckerausgabe

20D5 3A9B40      LD     A,(409BH)     :A = Drucker-POS
20D8 86          ADD    A,(HL)        :A = Drucker-POS + Stringlänge
20D9 FE84        CP     84H           :> 132 ?
20DB 1809        JR     20E6H         :Ja: neue Zeile beginnen

: Bildschirmausgabe

20DD 3A9D40      LD     A,(409DH)     :A = maximale Zeichenzahl pro
                                      :Zeile
20E0 47          LD     B,A           :B = A
20E1 3AA640      LD     A,(40A6H)     :A = Bildschirm-POS
20E4 86          ADD    A,(HL)        :A = Bildschirm-POS + Stringlänge
20E5 B8          CP     B             :> Zeilenlänge ?
20E6 D4FE20      CALL   NC.20FEH      :Ja: neue Zeile beginnen
```

```
; Cassettenausgabe

  20E9 CDAA28        CALL   28AAH        :String ausgeben
  20EC 3E20          LD     A,20H        :und Leerzeichen zur Trennung
  20EE CD2A03        CALL   032AH        :ausgeben
  20F1 B7            OR     A            :Z = 0 (nächsten Befehl nicht
                                         :ausführen)
  20F2 CCAA28        CALL   Z,28AAH      :String ausgeben wenn das
                                         :Argument im STR-Format war
  20F5 E1            POP    HL           :PTZ zurück
  20F6 C39B20        JP     209BH        :Nächstes Argument bearbeiten

; Neue Zeile beginnen

  20F9 3AA640        LD     A,(40A6H)    :A = Bildschirm-POS
  20FC B7            OR     A   -        :POS = 0 ? (Neue Zeile schon
                                         :begonnen ?)
  20FD C8            RET    Z            :Ja: Fertig

  20FE 3E0D          LD     A,0DH        :Nein: CR/LF ausgeben
  2100 CD2A03        CALL   032AH
  2103 CDD041        CALL   41D0H        :DOS
  2106 AF            XOR    A            :A = 0
  2107 C9            RET

; ',' bei PRINT:
; Cursor auf nächste TAB-Position setzen

  2108 CDD341        CALL   41D3H        :DOS
  210B 3A9C40        LD     A,(409CH)    :Ausgabe auf Cassette ?
  210E B7            OR     A
  210F F21921        JP     P,2119H      :Nein: weiter bei 2119H

; Cassettenausgabe

  2112 3E2C          LD     A,2CH        :Ja: ASCII-Zeichen ',' aus-
  2114 CD2A03        CALL   032AH        :geben
  2117 184B          JR     2164H        :Nächstes Argument bearbeiten

; PRINT ',' auf Bildschirm oder Drucker
; (Cursor auf nächste TAB-Position setzen)

  2119 2808          JR     Z,2123H      :weiter bei 2123H bei Bildschirm-
                                         :ausgabe

; Druckerausgabe

  211B 3A9B40        LD     A,(409BH)    :A = Drucker-POS
  211E FE70          CP     70H          :A > 112 ?
  2120 C32B21        JP     212BH        :Ja: neue Zeile beginnen
```

```
: Bildschirmausgabe

  2123 3A9E40        LD      A,(409EH)     ;A = höchste TAB-Position
  2126 47            LD      B,A           ;B = A
  2127 3AA640        LD      A,(40A6H)     ;A = Bildschirm-POS
  212A B8            CP      B             ;POS schon größer als höchste
                                           ;TAB-Position ?
  212B D4FE20        CALL    NC,20FEH      ;Ja: neue Zeile beginnen
  212E 3034          JR      NC,2164H      ;und nächstes Arg bearbeiten
  2130 D60A          SUB     0AH           ;Durch fortlaufende Subtraktion
  2132 30FC          JR      NC,2130H      ;eine Division durch 10
                                           ;durchführen
  2134 2F            CPL                   ;A = -A (da das Ergebnis der
                                           ;letzten Subtraktion < Null war.
                                           ;A ist jetzt der Rest der
                                           ;Division der aktuelle Cursor-
                                           ;position durch 10 und dies ist
                                           ;gleich dem Abstand zur nächsten
                                           ;Tabulatorposition)
  2135 1823          JR      215AH         ;A Leerzeichen ausgeben

: PRINT TAB(

  2137 CD1B2B        CALL    2B1BH         ;TAB-Arg nach DE holen
  213A CDB230        CALL    30B2H         ;E = neuer POS-Wert
  213D CF            RST     08H           ;Klammer geschlossen ?
  213E 29            DEFB    ')'
  213F 2B            DEC     HL            ;PTZ -1
  2140 E5            PUSH    HL            ;PTZ retten
  2141 CDD341        CALL    41D3H         ;DOS
  2144 3A9C40        LD      A,(409CH)     ;Cassettenausgabe ?
  2147 B7            OR      A
  2148 FA4A1E        JP      M,1E4AH       ;Ja: FC-Error
  214B CA5321        JP      Z,2153H       ;weiter bei 2153H bei Bildschirm-
                                           ;ausgabe

: Druckerausgabe

  214E 3A9B40        LD      A,(409BH)     ;A = Drucker-POS
  2151 1803          JR      2156H         ;weiter bei 2156H

: Bildschirmausgabe

  2153 3AA640        LD      A,(40A6H)     ;A = Bildschirm-POS
  2156 2F            CPL                   ;A = -A
  2157 83            ADD     A,E           ;A = -A + E = E - A
                                           ;(Neue POS - alte POS)
  2158 300A          JR      NC,2164H      ;Fertig wenn gewünschte
                                           ;TAB-Position schon erreicht
```

```
; A Leerzeichen ausgeben

   215A 3C            INC     A             :A +1
   215B 47            LD      B,A           :B = Zähler
   215C 3E20          LD      A,20H         :Leerzeichen ausgeben
   215E CD2A03        CALL    032AH
   2161 05            DEC     B             :Zähler -1
   2162 20FA          JR      NZ,215EH      :weiter bis B = 0

; ':' bei PRINT
; Keinen Zeilenvorschub generieren

   2164 E1            POP     HL            :PTZ zurück
   2165 D7            RST     10H           :nächstes Zeichen holen
   2166 C3A020        JP      20A0H         :nächstes Argument bearbeiten

; Nächste Ausgabe wieder zum Bildschirm leiten

   2169 3A9C40        LD      A,(409CH)     :--
   216C B7            OR      A
   216D 00            NOP
   216E 00            NOP
   216F 00            NOP
   2170 AF            XOR     A             :A = 0
   2171 329C40        LD      (409CH),A     :Ausgabeflag = 0
   2174 CDBE41        CALL    41BEH         :DOS
   2177 C9            RET

; Text '?REDO'

   2178 3F
   2179 52
   217A 45
   217B 44
   217C 4F
   217D 0D
   217E 00

; Daten bei INPUT oder READ nicht durch ',' getrennt

   217F 3ADE40        LD      A,(40DEH)     :Fehler in DATA-Zeile ?
   2182 B7            OR      A
   2183 C29119        JP      NZ,1991H      :Ja: SN-Error
   2186 CDB130        CALL    30B1H         :Nein: E auf korrekten TAB-Wert
                                            :prüfen (???)
   2189 B7            OR      A             :A = 0 ?  (???)
   218A 1E2A          LD      E,2AH         :E = Fehlercode für FD-Error
   218C CAA219        JP      Z,19A2H       :Ja: FD-Error
   218F C1            POP     BC            :Nein: PTZ vom Stack löschen
   2190 217821        LD      HL,2178H      :HL = Zeiger auf Text '?REDO'
   2193 CDA728        CALL    28A7H         :Text ausgeben
   2196 2AE640        LD      HL,(40E6H)    :HL = letzter PTZ-Stand
   2199 C9            RET                   :letzten Befehl wiederholen
```

```
; INPUT

  219A CD2828        CALL   2828H          ;ID-Error ?
  219D 7E            LD     A,(HL)         ;A=nächstes Zeichen
  219E CDD641        CALL   41D6H          ;DOS
  21A1 D623          SUB    23H            ;Zeichen = '#' ?
                                           ;(Ja: A = 00H wegen SUB !)
  21A3 32A940        LD     (40A9H),A      ;Flag = 0 wenn INPUT#
  21A6 7E            LD     A,(HL)         ;A = Zeichen
  21A7 2020          JR     NZ,21C9H       ;Nein: weiter bei 21C9H

; INPUT#

  21A9 CDAF35        CALL   35AFH          ;Zahl auswerten,
                                           ;Leader & Sync suchen
  21AC E5            PUSH   HL             ;PTZ retten
  21AD 06FA          LD     B,0FAH         ;B = 250 (Record-Länge)
  21AF 2AA740        LD     HL,(40A7H)     ;HL = Zeiger auf Zeilenbuffer
  21B2 CDED01        CALL   01EDH          ;Ein Byte lesen
  21B5 77            LD     (HL),A         ;und im Buffer ablegen
  21B6 23            INC    HL             ;Zeiger +1
  21B7 FE0D          CP     0DH            ;Recordende ?
  21B9 2802          JR     Z,21BDH        ;Ja: weiter bei 21BDH
  21BB 10F5          DJNZ   21B2H          ;Nein: Nächstes Byte lesen
  21BD 2B            DEC    HL             ;Zeiger -1
  21BE 3600          LD     (HL),00H       ;letztes Byte durch 00H ersetzen
  21C0 00            NOP                   ;--
  21C1 00            NOP
  21C2 00            NOP
  21C3 2AA740        LD     HL,(40A7H)     ;HL = Zeiger auf Zeilenbuffer
  21C6 2B            DEC    HL             ;Zeiger -1
  21C7 1822          JR     21EBH          ;Daten auswerten

; Normales INPUT

  21C9 01DB21        LD     BC,21DBH       ;RET-Adr auf 21DBH setzen
  21CC C5            PUSH   BC
  21CD FE22          CP     22H            ;Text vorher ausgeben ?
  21CF C0            RET    NZ             ;Nein: weiter bei 21BDH
  21D0 CD6628        CALL   2866H          ;Ja: Text übernehmen
  21D3 CF            RST    08H            ;Danach muß ';' folgen
  21D4 3B            DEFB   ';'
  21D5 E5            PUSH   HL             ;PTZ retten
  21D6 CDAA28        CALL   28AAH          ;Text ausgeben
  21D9 E1            POP    HL             ;PTZ zurück
  21DA C9            RET                   ;weiter bei 21BDH

  21DB E5            PUSH   HL             ;PTZ retten
  21DC CDB31B        CALL   1BB3H          ;'? ' ausgeben und zur Zeilen-
                                           ;eingabe springen
```

```
  21DF C1          POP    BC           ;BC = PTZ
  21E0 DABE1D      JP     C,1DBEH      ;Nach STOP springen wenn Break
                                       ;gedrückt wurde
  21E3 23          INC    HL           ;Zeiger +1
  21E4 7E          LD     A,(HL)       ;erstes Zeichen holen
  21E5 B7          OR     A            ;Ist es 00H (wurde nur 'RETURN'
                                       ;gedrückt ?)
  21E6 2B          DEC    HL           ;Zeiger -1
  21E7 C5          PUSH   BC           ;PTZ retten
  21E8 CA041F      JP     Z,1F04H      ;Ja: PTZ zur nächsten Anweisung
                                       ;erhöhen und Variablen
                                       ;unverändert lassen
  21EB 362C        LD     (HL),2CH     ;Nein: ',' als Trennung in den
                                       ;Buffer setzen
  21ED 1805        JR     21F4H        ;weiter bei 21F4H

; READ

  21EF E5          PUSH   HL           ;PTZ retten
  21F0 2AFF40      LD     HL,(40FFH)   ;HL = DATA-Zeiger
  21F3 F6AF        OR     0AFH         ;A <> 0 (bei READ)
 *21F4   AF        XOR    A            ;A = 0 (bei INPUT)
  21F5 32DE40      LD     (40DEH),A    ;Flag abspeichern
  21F8 E3          EX     (SP),HL      ;PTZ zurück, DATA-Zeiger retten
  21F9 1802        JR     21FDH        ;weiter bei 21FDH

; Nächste Daten übernehmen
; DATA-Zeiger = Zeiger auf die Daten
; PTZ = Zeiger auf die Variablen

  21FB CF          RST    08H          ;',' als Trennung ?
  21FC 2C          DEFB   2CH
  21FD CD0D26      CALL   260DH        ;Adresse der nach READ bzw. INPUT
                                       ;angegebenen Variablen ermitteln
  2200 E3          EX     (SP),HL      ;PTZ retten, DATA-Zeiger zurück
  2201 D5          PUSH   DE           ;Variablenadresse retten
  2202 7E          LD     A,(HL)       ;A = Datenzeichen
  2203 FE2C        CP     2CH          ;Trennzeichen gefunden ?
  2205 2826        JR     Z,222DH      ;Ja: weiter bei 222DH

; Kein Trennzeichen (',') gefunden

  2207 3ADE40      LD     A,(40DEH)    ;Wird gerade eine READ-
  220A B7          OR     A            ;Anweisung bearbeitet ?
  220B C29622      JP     NZ,2296H     ;Ja: DATA-Zeiger auf nächste
                                       ;DATA-Zeile erhöhen
  220E 3AA940      LD     A,(40A9H)    ;INPUT# ?
  2211 B7          OR     A
  2212 1E06        LD     E,06H        ;E = Fehlercode für OD-Error
  2214 CAA219      JP     Z,19A2H      ;Ja: Fehler erzeugen
  2217 3E3F        LD     A,3FH        ;Fragezeichen ausgeben
  2219 CD2A03      CALL   032AH
  221C CDB31B      CALL   1BB3H        ;und Eingabe fehlender Daten
                                       ;verlangen ( '?? ' )
```

```
221F D1                 POP     DE          ;Variablenadresse zurück
2220 C1                 POP     BC          ;PTZ zurück
2221 DABE1D             JP      C,1DBEH     ;Nach STOP springen wenn Break
                                            ;gedrückt wurde
2224 23                 INC     HL          ;Zeiger +1
2225 7E                 LD      A,(HL)      ;A = erstes Zeichen
2226 B7                 OR      A           ;wurde nur 'RETURN' gedrückt
                                            ;(erstes Zeichen = 00H)
2227 2B                 DEC     HL          ;Zeiger -1
2228 C5                 PUSH    BC          ;PTZ retten
2229 CA041F             JP      Z,1F04H     ;Ja: PTZ auf nächsten Befehl
                                            ;erhöhen und Variablen belassen
222C D5                 PUSH    DE          ;Nein: Variablenadresse retten

; Daten übernehmen
; HL = DATA-Zeiger auf Trennzeichen (',')

222D CDDC41             CALL    41DCH       ;DOS
2230 E7                 RST     20H         ;TSTTYP, Typ der Variablen, die
                                            ;die Daten übernehmen soll,
                                            ;testen
2231 F5                 PUSH    AF          ;VT - 3 retten
2232 2019               JR      NZ,224DH    ;Sprung bei numerischen Variablen

; Daten in Stringvariable übernehmen

2234 D7                 RST     10H         ;Erstes Zeichen nach A
2235 57                 LD      D,A
2236 47                 LD      B,A         ;D = B = erstes Zeichen
2237 FE22               CP      22H         ;Ist der String durch '"'
                                            ;begrenzt ?
2239 2805               JR      Z,2240H     ;Ja: B = D = Stringende
223B 163A               LD      D,3AH       ;Nein: Stringende kann ':' (3AH)
223D 062C               LD      B,2CH       ;oder ',' (2CH) sein
223F 2B                 DEC     HL          ;Zeiger -1
2240 CD6928             CALL    2869H       ;String übernehmen
2243 F1                 POP     AF          ;VT - 3 zurück
2244 EB                 EX      DE,HL       ;DE = DATA-Zeiger
2245 215A22             LD      HL,225AH    ;RET-Adr auf 225AH setzen
2248 E3                 EX      (SP),HL     ;HL = Variablenadresse
2249 D5                 PUSH    DE          ;DATA-Zeiger retten
224A C3331F             JP      1F33H       ;Neuen Wert in Variable kopieren
                                            ;und weiter bei 225AH

; Daten in numerische Variable übernehmen

224D D7                 RST     10H         ;Zeiger auf erstes Zeichen setzen
224E F1                 POP     AF          ;A = VT - 3
224F F5                 PUSH    AF
2250 014322             LD      BC,2243H    ;RET-Adr auf 2243H setzen
2253 C5                 PUSH    BC          ;(Wert in Variable kopieren)
2254 DA6C0E             JP      C,0E6CH     ;INT- oder SNG-Wert übernehmen
2257 D2650E             JP      NC,0E65H    ;DBL-Wert übernehmen
```

: Fortsetzung nach Datenübernahme

```
225A 2B          DEC   HL           :PTZ -1
225B D7          RST   10H          :Ende der Datenzeile erreicht ?
225C 2805        JR    Z,2263H      :Ja: weiter bei 2263H
225E FE2C        CP    2CH          :Nächste Daten mit ',' getrennt ?
2260 C27F21      JP    NZ,217FH     :Nein: Fehlerbehandlung bei 217FH
2263 E3          EX    (SP),HL      :DATA-Zeiger retten, HL = PTZ
2264 2B          DEC   HL           :PTZ -1
2265 D7          RST   10H          :READ- bzw. INPUT-Anweisung be-
                                    :endet ?
2266 C2FB21      JP    NZ,21FBH     :Nein: nächste Variable
                                    :bearbeiten
2269 D1          POP   DE           :Ja: DATA-Zeiger zurück
226A 00          NOP                :--
226B 00          NOP
226C 00          NOP
226D 00          NOP
226E 00          NOP
226F 3ADE40      LD    A,(40DEH)    :Wurde eine READ-Anweisung
                                    :bearbeitet ?
2272 B7          OR    A
2273 EB          EX    DE,HL        :DE = PTZ, HL = DATA-Zeiger
2274 C2961D      JP    NZ,1D96H     :Ja: DATA-Zeiger zurückschreiben
2277 D5          PUSH  DE           :Nein: PTZ retten
2278 CDDF41      CALL  41DFH        :DOS
227B B6          OR    (HL)         :Noch Daten im Buffer ?
227C 218622      LD    HL,2286H     :HL : Text '?Extra ignored'
227F C4A728      CALL  NZ,28A7H     :Ja: Text ausgeben
2282 E1          POP   HL           :PTZ zurück
2283 C36921      JP    2169H        :Nächste Ausgabe auf Bildschirm
```

: Text '?Extra ignored'

```
2286 3F
2287 45
2288 78
2289 74
228A 72
228B 61
228C 20
228D 69
228E 67
228F 6E
2290 6F
2291 72
2292 65
2293 64
2294 0D
2295 00
```

```
: Keine Datentrennung in der aktuellen DATA-Zeile gefunden
: HL auf nächste DATA-Zeile erhöhen

  2296 CD051F        CALL   1F05H        :HL bis zum Befehls- bzw.
                                         :Zeilenende erhöhen
  2299 B7            OR     A            :Zeilenende erreicht ?
  229A 2012          JR     NZ,22AEH     :Nein: weiter bei 22AEH
  229C 23            INC    HL           :Ja: Programmende erreicht ?
  229D 7E            LD     A,(HL)       :(Zeilenpointer = 0000H ?)
  229E 23            INC    HL
  229F B6            OR     (HL)
  22A0 1E06          LD     E,06H        :E = Fehlercode für OD-Error
  22A2 CAA219        JP     Z,19A2H      :Ja: Fehler erzeugen
  22A5 23            INC    HL           :Nein: DE = Zeilennummer
  22A6 5E            LD     E,(HL)
  22A7 23            INC    HL
  22A8 56            LD     D,(HL)
  22A9 EB            EX     DE,HL        :Zeilennummer der DATA-Zeile
  22AA 22DA40        LD     (40DAH),HL   :abspeichern
  22AD EB            EX     DE,HL
  22AE D7            RST    10H          :Wurde eine DATA-Zeile gefunden ?
  22AF FE88          CP     88H          :Ist das erste Zeichen ein
                                         :'DATA'-Token ?
  22B1 20E3          JR     NZ,2296H     :Nein: weitersuchen
  22B3 C32D22        JP     222DH        :Ja: HL = neuer DATA-Zeiger

: NEXT

  22B6 110000        LD     DE,0000H     :Defaultadresse, wenn keine
                                         :Variable angegeben wurde
  22B9 C40D26        CALL   NZ,260DH     :Adresse ermitteln, wenn ein
                                         :Variablenname angegeben wurde
  22BC 22DF40        LD     (40DFH),HL   :PTZ abspeichern
  22BF CD3619        CALL   1936H        :FOR-Stack mit VARPTR = DE suchen
                                         :(nächstes FOR-Stack wenn DE = 0)
  22C2 C29D19        JP     NZ,199DH     :NF-Error wenn FOR-Stack nicht
                                         :gefunden
  22C5 F9            LD     SP,HL        :SP = FOR-Stack Zeiger
  22C6 22E840        LD     (40E8H),HL   :SP-Wert abspeichern
  22C9 D5            PUSH   DE           :Variablenadresse retten
  22CA 7E            LD     A,(HL)       :A = Step-SGN
  22CB 23            INC    HL           :Zeiger +1
  22CC F5            PUSH   AF           :Step-SGN retten
  22CD D5            PUSH   DE           :Variablenadresse retten
  22CE 7E            LD     A,(HL)       :Variablentyp holen
  22CF 23            INC    HL           :Zeiger +1
  22D0 B7            OR     A            :Schleifenvariable Integer ?
  22D1 FAEA22        JP     M,22EAH      :Ja: weiter bei 22EAH
```

```
; NEXT für SNG-Variable

  22D4 CDB109          CALL    09B1H       :X = BCDE = (HL) (Stepwert nach X
                                           :holen)
  22D7 E3              EX      (SP),HL     :Zeiger retten, HL = VARPTR
  22D8 E5              PUSH    HL          :Adresse der Schleifenvariablen
                                           :retten
  22D9 CD0B07          CALL    070BH       :X = X + (HL) (Schleifenvariable
                                           :um Stepwert erhöhen)
  22DC E1              POP     HL          :Adresse zurück
  22DD CDCB09          CALL    09CBH       :(HL) = X (neuen Wert in Variable
                                           :kopieren)
  22E0 E1              POP     HL          :Zeiger zurück (Zeigt jetzt auf
                                           :den Endwert)
  22E1 CDC209          CALL    09C2H       :BCDE = (HL) = Endwert
  22E4 E5              PUSH    HL          :Zeiger retten
  22E5 CD0C0A          CALL    0A0CH       :CP X,BCDE (Schleifenvariable mit
                                           :Endwert vergleichen)
  22E8 1829            JR      2313H       :weiter bei 2313H

; NEXT für INT-Variable

  22EA 23              INC     HL          :Die ersten 4 Bytes des FOR-
  22EB 23              INC     HL          :Stacks sind unbenutzt
  22EC 23              INC     HL
  22ED 23              INC     HL
  22EE 4E              LD      C,(HL)      :BC = Stepwert
  22EF 23              INC     HL
  22F0 46              LD      B,(HL)
  22F1 23              INC     HL
  22F2 E3              EX      (SP),HL     :Zeiger retten, HL = Adresse
  22F3 5E              LD      E,(HL)      :DE = Schleifenwert
  22F4 23              INC     HL
  22F5 56              LD      D,(HL)
  22F6 E5              PUSH    HL          :Adresse+1 retten
  22F7 69              LD      L,C         :HL = Stepwert
  22F8 60              LD      H,B
  22F9 CDD20B          CALL    0BD2H       :X = HL = HL + DE (Schleifen-
                                           :variable um Stepwert erhöhen)
  22FC 3AAF40          LD      A,(40AFH)   :Überlauf ?
  22FF FE04            CP      04H         :Wurde der VT auf SNG geändert ?
  2301 CAB207          JP      Z,07B2H     :Ja: OV-Error
  2304 EB              EX      DE,HL       :DE = Neuer Schleifenwert
  2305 E1              POP     HL          :Adresse+1 zurück
  2306 72              LD      (HL),D      :Neuen Schleifenwert zurück-
  2307 2B              DEC     HL          :schreiben
  2308 73              LD      (HL),E
  2309 E1              POP     HL          :Zeiger zurück
  230A D5              PUSH    DE          :Schleifenwert retten
  230B 5E              LD      E,(HL)      :DE = Endwert
  230C 23              INC     HL
  230D 56              LD      D,(HL)
  230E 23              INC     HL
  230F E3              EX      (SP),HL     :Zeiger retten, HL = Schleifen-
                                           :wert
```

```
2310 CD390A          CALL    0A39H        :CP HL,DE (Schleifenwert mit End-
                                          :wert vergleichen)
2313 E1              POP     HL           :Zeiger zurück
2314 C1              POP     BC           :B = Step-SGN:
                                          :B = 01H: Schleife beendet wenn
                                          :         Schleifenwert > Endwert
                                          :B = FFH: Schleife beendet wenn
                                          :         Schleifenwert < Endwert
                                          :
                                          :A = FFH: Schleifenwert < Endwert
                                          :A = 00H: Schleifenwert = Endwert
                                          :A = 01H: Schleifenwert > Endwert
2315 90              SUB     B            :Schleife beendet ?
2316 CDC209          CALL    09C2H        :BCDE = (HL)
                                          :(BC = ZP auf Schleifenanfang,
                                          :DE = Zeilennummer)
2319 2809            JR      Z,2324H      :Ja: weiter bei 2324H

: Schleife nicht beendet

231B EB              EX      DE,HL        :HL = Zeilennummer
231C 22A240          LD      (40A2H),HL   :Als aktuelle ZN abspeichern
231F 69              LD      L,C          :HL = Zeilenpointer
2320 60              LD      H,B
2321 C31A1D          JP      1D1AH        :FOR-Stack wieder aufbauen und
                                          :Programm ab (HL) fortsetzen

: Schleife beendet

2324 F9              LD      SP,HL        :SP = Stackende (aktuelles
                                          :FOR-Stack gelöscht)
2325 22E840          LD      (40E8H),HL   :Neuen SP abspeichern
2328 2ADF40          LD      HL,(40DFH)   :HL = PTZ
232B 7E              LD      A,(HL)       :nächstes Zeichen holen
232C FE2C            CP      2CH          :Ist es ein Komma
232E C21E1D          JP      NZ,1D1EH     :Nein: Programm ab (HL) fort-
                                          :setzen
2331 D7              RST     10H          :Ja: PTZ auf nächsten Variablen-
                                          :namen erhöhen
2332 CDB922          CALL    22B9H        :und nächste Variable bearbeiten
```

```
; Klammerausdruck bearbeiten und Ergebnis in X ablegen

  2335 CF             RST    08H           :PTZ muß auf '(' stehen
  2336 28             DEFB   '('

; Ausdruck bearbeiten und Ergebnis in X ablegen

  2337 2B             DEC    HL            :PTZ -1
  2338 1600           LD     D,00H         :Prioritätsflag = 0
  233A D5             PUSH   DE            :Flag retten
  233B 0E01           LD     C,01H         :Speicherprüfung: noch 2 Bytes
  233D CD6319         CALL   1963H         :frei ?
  2340 CD9F24         CALL   249FH         :1. Argument holen
  2343 22F340         LD     (40F3H),HL    :PTZ abspeichern
  2346 2AF340         LD     HL,(40F3H)    :PTZ zurückholen
  2349 C1             POP    BC -          :Prioritätsflag zurück
  234A 7E             LD     A,(HL)        :A = Token nach Argument
                                           :(Operatorcode)
  234B 1600           LD     D,00H         :Operatorflag = 0
  234D D6D4           SUB    0D4H          :Vergleichsoperator ?
                                           :( < , = , > )
  234F 3813           JR     C,2364H       :Nein: weiter bei 2364H
  2351 FE03           CP     03H           :(Tokenwert zwischen D4H und
                                           :D7H ?)
  2353 300F           JR     NC,2364H      :Nein: weiter bei 2364H

: Vergleichsoperatoren verarbeiten ( < , = , > , =< , >= , <= , => , <> )

  2355 FE01           CP     01H           :CY = 1 wenn '<' gefunden
  2357 17             RLA                  :'<' : A = 01 (Bit 0 gesetzt)
                                           :'=' : A = 02 (Bit 1 gesetzt)
                                           :'>' : A = 04 (Bit 2 gesetzt)
  2358 AA             XOR    D             :A mit letztem Operator ver-
                                           :knüpfen, A = 00 wenn zweimal
                                           :der gleiche Operator angegeben
                                           :wurde
  2359 BA             CP     D             :CY = 1 wenn A = 0 !!
  235A 57             LD     D,A           :D = neuer Operatorcode
  235B DA9719         JP     C,1997H       :SN-Error wenn Operator doppelt
                                           :angegeben
  235E 22D840         LD     (40D8H),HL    :PTZ abspeichern
  2361 D7             RST    10H           :Nächsten Operator holen
  2362 18E9           JR     234DH         :Noch ein Vergleichsoperator ?
```

: Kein Vergleichsoperator gefunden

```
 2364 7A              LD      A,D          ;A = Operatorflag
 2365 B7              OR      A            ;Wurden Vergleichsoperatoren
                                           ;gefunden ?
 2366 C2EC23          JP      NZ,23ECH     ;Ja: weiter bei 23ECH
 2369 7E              LD      A,(HL)       ;Nein: A = Operatorcode
 236A 22D840          LD      (40D8H),HL   ;PTZ abspeichern
 236D D6CD            SUB     0CDH         ;Verknüpfungscode gefunden ?
 236F D8              RET     C            ;Nein: Fertig
 2370 FE07            CP      07H          ;( +, -, * , /,
                                           ;Hochpfeil, AND,  OR) ?
 2372 D0              RET     NC           ;Nein: Fertig
 2373 5F              LD      E,A          ;DE = Codeoffset (0='+', 6='OR')
 2374 3AAF40          LD      A,(40AFH)    ;A = VT (des 1. Arguments)
 2377 D603            SUB     03H -        ;Stringargument ?
 2379 B3              OR      E            ;und Code = '+' ?
 237A CA8F29          JP      Z,298FH      ;Ja: Stringaddition bei 298FH
```

: Prioritätsprüfung

```
 237D 219A18          LD      HL,189AH     ;HL = Zeiger auf Prioritäts-
                                           ;tabelle
 2380 19              ADD     HL,DE        ;Offset addieren
 2381 78              LD      A,B          ;A = letzte Priorität
                                           ;(Ist am Anfang = 0 !)
 2382 56              LD      D,(HL)       ;D = aktuelle Priorität
 2383 BA              CP      D            ;Beide Prioritätswert vergleichen
 2384 D0              RET     NC           ;Zwischenergebnis zuerst be-
                                           ;rechnen wenn der alte Operator
                                           ;höhere Priorität hatte
 2385 C5              PUSH    BC           ;Letzte Priorität retten
 2386 014623          LD      BC,2346H     ;RET-Adr auf 2346H setzen
 2389 C5              PUSH    BC
 238A 7A              LD      A,D          ;A = neue Priorität
 238B FE7F            CP      7FH          ;Potenzberechnung ?
 238D CAD423          JP      Z,23D4H      ;Ja: weiter bei 23D4H
 2390 FE51            CP      51H          ;Logische Verknüpfung (AND, OR) ?
 2392 DAE123          JP      C,23E1H      ;Ja: weiter bei 23E1H
```

```
; Operatoren '+' , '-' , '*' und '/' bearbeiten
; 1. Argument (bzw. Zwischenergebnis) im Stack ablegen

  2395 212141         LD      HL,4121H      ;HL = Zeiger auf X
  2398 B7             OR      A             ;CY = 0
  2399 3AAF40         LD      A,(40AFH)     ;A = VT
  239C 3D             DEC     A             ;A -3
  239D 3D             DEC     A             ;Ist X im STR-Format ?
  239E 3D             DEC     A
  239F CAF60A         JP      Z,0AF6H       ;Ja: TM-Error (Für Strings ist
                                            ;nur '+' erlaubt und dieser
                                            ;Operator wurde bereits
                                            ;abgefragt)
  23A2 4E             LD      C,(HL)        ;BC = INT-Wert
  23A3 23             INC     HL
  23A4 46             LD      B,(HL)
  23A5 C5             PUSH    BC            ;INT-Wert ins Stack
  23A6 FAC523         JP      M,23C5H       ;Fertig wenn X im INT-Format
  23A9 23             INC     HL            ;BC = MSBs des SNG-Werts
  23AA 4E             LD      C,(HL)
  23AB 23             INC     HL
  23AC 46             LD      B,(HL)
  23AD C5             PUSH    BC            ;MSBs des SNG-Werts ins Stack
  23AE F5             PUSH    AF            ;VT-3 retten
  23AF B7             OR      A             ;X im SNG-Format ? (CY = 0)
  23B0 E2C423         JP      PO,23C4H      ;Ja: Fertig
  23B3 F1             POP     AF            ;Nein: VT-3 zurück
  23B4 23             INC     HL
  23B5 3803           JR      C,23BAH       ;Sprung wird nie ausgeführt
  23B7 211D41         LD      HL,411DH      ;LSBs des DBL-Werts ins Stack
  23BA 4E             LD      C,(HL)        ;retten
  23BB 23             INC     HL
  23BC 46             LD      B,(HL)
  23BD 23             INC     HL
  23BE C5             PUSH    BC
  23BF 4E             LD      C,(HL)
  23C0 23             INC     HL
  23C1 46             LD      B,(HL)
  23C2 C5             PUSH    BC
  23C3 06F1           LD      B,0F1H        ;--
 *23C4   F1           POP     AF            ;VT-3 zurück (bei INT- oder SNG-
                                            ;Wert)
  23C5 C603           ADD     A,03H         ;A +3
  23C7 4B             LD      C,E           ;C = Offset des Operatorcodes
  23C8 47             LD      B,A           ;B = VT
  23C9 C5             PUSH    BC            ;Beides ins Stack retten
  23CA 010624         LD      BC,2406H      ;RET-Adr auf 2406H setzen
  23CD C5             PUSH    BC
  23CE 2AD840         LD      HL,(40D8H)    ;PTZ zurück nach HL
  23D1 C33A23         JP      233AH         ;Nächstes Argument bearbeiten
```

; Potenzfunktion berechnen

```
23D4 CDB10A        CALL    0AB1H       ;X = CSNG (X)
23D7 CDA409        CALL    09A4H       ;(SP) = X
23DA 01F213        LD      BC,13F2H    ;BC = Adresse der Potenzfunktion
23DD 167F          LD      D,7FH       ;D = Prioritätscode
23DF 18EC          JR      23CDH       ;2. Argument holen und Funktion
                                       ;berechnen
```

: Logische Verknüpfung bearbeiten

```
23E1 D5            PUSH    DE          ;Prioritätscode retten
23E2 CD7F0A        CALL    0A7FH       ;HL = X = CINT(X)
23E5 D1            POP     DE          ;Prioritätscode zurück
23E6 E5            PUSH    HL          ;1. Argument retten
23E7 01E925        LD      BC,25E9H    ;BC = Adresse der AND/OR Routine
23EA 18E1          JR      23CDH       ;2. Argument holen und Ver-
                                       ;knüpfung bearbeiten
```

: Vergleichsoperatoren bearbeiten

```
23EC 78            LD      A,B         ;A = Prioritätscode
23ED FE64          CP      64H         ;Hatte der letzte Operator eine
                                       ;höhere Priorität ?
23EF D0            RET     NC          ;Ja: Zwischenergebnis berechnen
23F0 C5            PUSH    BC          ;Nein: Prioritätscode retten
23F1 D5            PUSH    DE          ;Operatorcode retten
23F2 110464        LD      DE,6404H    ;D = aktuelle Priorität,
                                       ;E = Offset für Grundrechenarten
                                       ;(siehe Tabellen ab 18ABH)
23F5 21B825        LD      HL,25B8H    ;RET-Adr auf 25B8H setzen
23F8 E5            PUSH    HL
23F9 E7            RST     20H         ;TSTTYP, String ?
23FA C29523        JP      NZ,2395H    ;Nein: Zwischenergebnis ins Stack
                                       ;retten
23FD 2A2141        LD      HL,(4121H)  ;Ja: HL = Vektoradresse
2400 E5            PUSH    HL          ;Adresse retten
2401 018C25        LD      BC,258CH    ;RET-Adr auf 258CH (String-
                                       ;vergleich) setzen
2404 18C7          JR      23CDH       ;2. Argument holen und Funktion
                                       ;ausführen
```

```
: Zwischenergebnis berechnen
: 1. Wert ist im Stack, 2. Wert ist in X

  2406 C1            POP    BC           ;VT des 1. Arguments zurück
  2407 79            LD     A,C          ;A = Offset des Operatorcodes
  2408 32B040        LD     (40B0H),A    ;Codeoffset abspeichern
  240B 78            LD     A,B          ;A = VT
  240C FE08          CP     08H          ;1. Argument im DBL-Format ?
  240E 2828          JR     Z,2438H      ;Ja: weiter bei 2438H
  2410 3AAF40        LD     A,(40AFH)    ;A = VT des 2. Arguments
  2413 FE08          CP     08H          ;2. Argument im DBL-Format ?
  2415 CA6024        JP     Z,2460H      ;Ja: weiter bei 2460H
  2418 57            LD     D,A          ;D = VT des 2. Arguments
  2419 78            LD     A,B          ;A = VT des 1. Arguments
  241A FE04          CP     04H          ;1. Argument im SNG-Format ?
  241C CA7224        JP     Z,2472H      ;Ja: weiter bei 2472H
  241F 7A            LD     A,D          ;A = VT des 2. Arguments
  2420 FE03          CP     03H          ;2. Argument im STR-Format ?
  2422 CAF60A        JP     Z,0AF6H      ;Ja: TM-Error
  2425 D27C24        JP     NC,247CH     ;weiter bei 247CH wenn das
                                         ;2. Argument im SNG-Format ist

: 1. oder 2. Argument im INT-Format

  2428 21BF18        LD     HL,18BFH     ;HL = Zeiger auf die Sprung-
                                         ;tabelle der Grundrechenarten
                                         ;im INT-Format
  242B 0600          LD     B,00H        ;BC = Operatoroffset
  242D 09            ADD    HL,BC        ;Offset 2 mal addieren
  242E 09            ADD    HL,BC        ;(2 Byte Adresse)
  242F 4E            LD     C,(HL)       ;Adresse nach BC laden
  2430 23            INC    HL
  2431 46            LD     B,(HL)
  2432 D1            POP    DE           ;DE = 1. Argument
  2433 2A2141        LD     HL,(4121H)   ;Hl = 2. Argument
  2436 C5            PUSH   BC           ;Adresse ins Stack
  2437 C9            RET                 ;und Routine ausführen

: 1. Argument im DBL-Format

  2438 CDDB0A        CALL   0ADBH        ;X = CDBL(X) (2. Argument ins
                                         ;DBL-Format umwandeln)
  243B CDFC09        CALL   09FCH        ;Y = X (2. Argument nach Y)
  243E E1            POP    HL           ;1. Argument aus dem Stack holen
  243F 221F41        LD     (411FH),HL   ;und nach X laden
  2442 E1            POP    HL
  2443 221D41        LD     (411DH),HL
  2446 C1            POP    BC
  2447 D1            POP    DE
  2448 CDB409        CALL   09B4H        ;X = BCDE
  244B CDDB0A        CALL   0ADBH        ;X = CDBL(X) (1. Argument ins
                                         ;DBL-Format umwandeln)
```

```
244E 21AB18            LD      HL,18ABH      :HL = Zeiger auf die Sprung-
                                             :tabelle der Grundrechenarten
                                             :im DBL-Format
2451 3AB040            LD      A,(40B0H)     :A = Operatorcodeoffset
2454 07                RLCA                  : * 2 (2 Bytes Adresse)
2455 C5                PUSH    BC            :BC retten
2456 4F                LD      C,A           :BC = Tabellenoffset
2457 0600              LD      B,00H
2459 09                ADD     HL,BC         :Offset addieren
245A C1                POP     BC            :BC zurück
245B 7E                LD      A,(HL)        :HL = Routinenadresse
245C 23                INC     HL
245D 66                LD      H,(HL)
245E 6F                LD      L,A
245F E9                JP      (HL)          :Routine ausführen

: 2. Argument im DBL-Format

2460 C5                PUSH    BC            :VT um Offset retten
2461 CDFC09            CALL    09FCH         :Y = X (2. Argument nach Y)
2464 F1                POP     AF            :VT zurück
2465 32AF40            LD      (40AFH),A     :VT abspeichern
2468 FE04              CP      04H           :1. Argument im SNG-Format ?
246A 28DA              JR      Z,2446H       :Ja: 1. Argument vom Stack holen
                                             :und ins DBL-Format bringen
246C E1                POP     HL            :Nein: 1. Argument ist im
                                             :INT-Format: HL = 1. Argument
246D 222141            LD      (4121H),HL    :1. Argument nach X bringen
2470 18D9              JR      244BH         :und ins DBL-Format umwandeln

: 1. Argument im SNG-Format

2472 CDB10A            CALL    0AB1H         :X = CSNG(X) (2. Argument ins
                                             :SNG-Format umwandeln)
2475 C1                POP     BC            :1. Argument aus dem Stack
2476 D1                POP     DE            :holen
2477 21B518            LD      HL,18B5H      :HL = Zeiger auf die Sprung-
                                             :tabelle
247A 18D5              JR      2451H         :Funktion berechnen

: 2. Argument im SNG-Format, 1. Argument im INT-Format

247C E1                POP     HL            :1. Argument zurück
247D CDA409            CALL    09A4H         :(SP) = X (2. Argument ins Stack
                                             :retten)
2480 CDCF0A            CALL    0ACFH         :X = CSNG(HL)
2483 CDBF09            CALL    09BFH         :BCDE = X = 1. Argument
2486 E1                POP     HL            :1. Argument aus dem Stack
2487 222341            LD      (4123H),HL    :nach X bringen
248A E1                POP     HL
248B 222141            LD      (4121H),HL
248E 18E7              JR      2477H         :weiter bei 2477H
```

```
; IDIV: X = DE / HL
; Dividiert zwei Integer-Zahlen mit Ergebnis im SNG-Format
; I: DE = Dividend
;    HL = Divisor
; O: X  = Quotient

  2490 E5             PUSH   HL          ;Divisor retten
  2491 EB             EX     DE,HL       ;HL = Dividend
  2492 CDCF0A         CALL   0ACFH       ;X = CSNG(DE) (X = Dividend)
  2495 E1             POP    HL          ;Dividend zurück
  2496 CDA409         CALL   09A4H       ;(SP) = X (Dividend in Stack)
  2499 CDCF0A         CALL   0ACFH       ;X = CSNG(HL) (X = Divisor)
  249C C3A008         JP     08A0H       ;X = (SP) / X (SDIV ausführen)

; Argument bei (PTZ) decodieren und Ergebnis in X ablegen

  249F D7             RST    10H         ;Operand angegeben ?
  24A0 1E28           LD     E,28H       ;E = Fehlercode für MO-Error
  24A2 CAA219         JP     Z,19A2H     ;Nein: MO-Error
  24A5 DA6C0E         JP     C,0E6CH     ;Ja: Ziffer angegeben ?
                                         ;Ja: Zahl decodieren und Ergebnis
                                         ;in X ablegen
  24A8 CD3D1E         CALL   1E3DH       ;Nein: Variable angegeben ?
  24AB D24025         JP     NC,2540H    ;Ja: weiter bei 2540H
  24AE FECD           CP     0CDH        ;'+'-Token ? (Positives
                                         ;Vorzeichen) ?
  24B0 28ED           JR     Z,249FH     ;Ja: Übergehen, nächstes Zeichen
                                         ;holen
  24B2 FE2E           CP     2EH         ;'.' (Dezimalpunkt) ?
  24B4 CA6C0E         JP     Z,0E6CH     ;Ja: Zahl decodieren
  24B7 FECE           CP     0CEH        ;'-'-Token ? (negatives
                                         ;Vorzeichen) ?
  24B9 CA3225         JP     Z,2532H     ;Ja: weiter bei 2532H
  24BC FE22           CP     22H         ;'"' (Stringkonstante) ?
  24BE CA6628         JP     Z,2866H     ;Ja: weiter bei 2866H
  24C1 FECB           CP     0CBH        ;'NOT'-Token ?
  24C3 CAC425         JP     Z,25C4H     ;Ja: weiter bei 25C4H
  24C6 FE26           CP     26H         ;'&' ? (Hexadezimale Konstante) ?
  24C8 CAE334         JP     Z,34E3H     ;Ja: weiter bei 34E3H
  24CB FEC3           CP     0C3H        ;'ERR'-Token ?
  24CD 200A           JR     NZ,24D9H    ;Nein: weiter bei 24D9H

; ERR

  24CF D7             RST    10H         ;PTZ erhöhen
  24D0 3A9A40         LD     A,(409AH)   ;A = Letzter Fehlercode
  24D3 E5             PUSH   HL          ;PTZ retten
  24D4 CDF827         CALL   27F8H       ;X = HL = A (ohne Vorzeichen)
  24D7 E1             POP    HL          ;PTZ zurück
  24D8 C9             RET

; Nicht 'ERR'

  24D9 FEC2           CP     0C2H        ;'ERL'-Token ?
  24DB 200A           JR     NZ,24E7H    ;Nein: weiter bei 24E7H
```

```
; ERL

 24DD D7         RST   10H          ;PTZ erhöhen
 24DE E5         PUSH  HL           ;PTZ retten
 24DF 2AEA40     LD    HL,(40EAH)   ;HL = ZN der letzten Fehlerzeile
 24E2 CD660C     CALL  0C66H        ;X = HL (SNG) (ohne Vorzeichen)
 24E5 E1         POP   HL           ;PTZ zurück
 24E6 C9         RET

; Nicht 'ERL'

 24E7 FEC0       CP    0C0H         ;'VARPTR'-Token ?
 24E9 2014       JR    NZ,24FFH     ;Nein: weiter bei 24FFH

; VARPTR

 24EB D7         RST   10H          ;PTZ erhöhen
 24EC CF         RST   08H          ;Jetzt muß '(' folgen
 24ED 28         DEFB  '('
 24EE CD0D26     CALL  260DH        ;Adresse der angegebenen
                                    ;Variablen nach DE laden.
                                    ;DE = 0000H wenn die Variable
                                    ;nicht gefunden wurde
 24F1 CF         RST   08H          ;Klammer geschlossen ?
 24F2 29         DEFB  ')'
 24F3 E5         PUSH  HL           ;PTZ retten
 24F4 EB         EX    DE,HL        ;HL = Variablenadresse
 24F5 7C         LD    A,H          ;Adresse = 0000H ?
 24F6 B5         OR    L
 24F7 CA4A1E     JP    Z,1E4AH      ;Ja: FC-Error
 24FA CD9A0A     CALL  0A9AH        ;Nein: X = HL (INT)
 24FD E1         POP   HL           ;PTZ zurück
 24FE C9         RET

; Nicht 'VARPTR'

 24FF FEC1       CP    0C1H         ;'USR'-Token ?
 2501 C37A3F     JP    3F7AH        ;Colour-Basic-Token abfangen
                                    ;weiter bei 27FEH wenn
                                    ;'USR'-Token gefunden
 2504 FEC5       CP    0C5H         ;'INSTR'-Token ?
 2506 CA9D41     JP    Z,419DH      ;Ja: weiter bei 419DH
 2509 FEC8       CP    0C8H         ;'MEM'-Token ?
 250B CAC927     JP    Z,27C9H      ;Ja: weiter bei 27C9H
 250E FEC7       CP    0C7H         ;'TIME$'-Token ?
 2510 CA7641     JP    Z,4176H      ;Ja: weiter bei 4176H
 2513 FEC6       CP    0C6H         ;'CHECK'-Token ?
 2515 CA3201     JP    Z,0132H      ;Ja: weiter bei 0132H
 2518 FEC9       CP    0C9H         ;'INKEY$'-Token ?
 251A CA9D01     JP    Z,019DH      ;Ja: weiter bei 019DH
 251D FEC4       CP    0C4H         ;'STRING$'-Token ?
 251F CA2F2A     JP    Z,2A2FH      ;Ja: weiter bei 2A2FH
 2522 FEBE       CP    0BEH         ;'FN'-Token ?
 2524 CA5541     JP    Z,4155H      ;Ja: weiter bei 4155H
```

```
2527 D6D7          SUB    0D7H         :Funktionstoken gefunden ?
2529 D24E25        JP     NC.254EH     :Ja: weiter bei 254EH
252C CD3523        CALL   2335H        :Nein: Es kann nur ein Klammer-
                                       :ausdruck bei (PTZ) stehen, alle
                                       :anderen Möglichkeiten wurden
                                       :bereits abgefangen
252F CF            RST    08H          :Klammer geschlossen ?
2530 29            DEFB   ')'
2531 C9            RET
```

; Negatives Vorzeichen verarbeiten

```
2532 167D          LD     D.7DH        :D = Prioritätscode (nächste
                                       :Priorität nach Potenzfunktion)
2534 CD3A23        CALL   233AH        :Argument nach X holen
2537 2AF340        LD     HL.(40F3H)   :PTZ holen
253A E5            PUSH   HL           :und retten
253B CD7B09        CALL   097BH        :Ergebnis typrichtig negieren
253E E1            POP    HL           :PTZ zurück
253F C9            RET
```

; Variable als Operand verarbeiten

```
2540 CD0D26        CALL   260DH        :Adresse der Variablen ermitteln
2543 E5            PUSH   HL           :PTZ retten
2544 EB            EX     DE.HL        :HL = Adresse
2545 222141        LD     (4121H).HL   :Adresse abspeichern
2548 E7            RST    20H          :TSTTYP, String ?
2549 C4F709        CALL   NZ.09F7H     :Nein: X = (HL) (Typrichtig)
254C E1            POP    HL           :PTZ zurück
254D C9            RET
```

; Funktionstoken gefunden
; A = 00H für SGN, 01H für INT ... 22H für RIGHT$ und 23H für MID$
; (siehe Keyword- und Adressentabelle 1608H ff)

```
254E 0600          LD     B.00H        :MSB des Offsets = 00H
2550 07            RLCA                :A = A * 2 (2 Bytes Adresse)
2551 4F            LD     C.A          :BC = Offset für Adressentabelle
2552 C5            PUSH   BC           :Offset retten
2553 D7            RST    10H          :PTZ erhöhen
2554 79            LD     A.C          :A = Offset-LSB
2555 FE41          CP     41H          :LSB > 41H ?
2557 3816          JR     C.256FH      :Nein: weiter bei 256FH
```

: 1. Argument von LEFT$, RIGHT$ oder MID$ bearbeiten

```
2559 CD3523          CALL   2335H        ;Klammerausdruck bearbeiten
255C CF              RST    08H          ;Nach dem 1. Argument muß ein
255D 2C              DEFB   ','          ;Komma folgen
255E CDF40A          CALL   0AF4H        ;TM-Error wenn das 1. Argument
                                         ;kein String war
2561 EB              EX     DE,HL        ;DE = PTZ
2562 2A2141          LD     HL,(4121H)   ;HL = Adresse des 1. Arg
2565 E3              EX     (SP),HL      ;Adresse retten
2566 E5              PUSH   HL           ;Offset zurück ins Stack
2567 EB              EX     DE,HL        ;HL = PTZ
2568 CD1C2B          CALL   2B1CH        ;2. Argument (Zahl) nach DE holen
256B EB              EX     DE,HL        ;DE = PTZ, HL = 2. Argument
256C E3              EX     (SP),HL      ;2. Argument retten, HL = Offset
256D 1814            JR     2583H        ;weiter bei 2583H
```

: Funktionsargument holen und Funktion ausführen

```
256F CD2C25          CALL   252CH        ;Klammerausdruck bearbeiten
2572 E3              EX     (SP),HL      ;PTZ retten, HL = Offset
2573 7D              LD     A,L          ;A = LSB des Offsets
2574 FE0C            CP     0CH          ;LSB < 0CH ?
                                         ;(SGN/INT/ABS/FRE/INP/POS) ?
2576 3807            JR     C,257FH      ;Ja: Funktion ausführen
2578 FE1B            CP     1BH          ;Nein: LSB < 1BH ? (SQR/RND/LOG
                                         ;EXP/COS/SIN/TAN/ATN) ?
257A E5              PUSH   HL           ;Offset retten
257B DCB10A          CALL   C,0AB1H      ;Ja: X = CSNG (X)
257E E1              POP    HL           ;Offset zurück
257F 113E25          LD     DE,253EH     ;RET-Adr auf 253EH setzen
2582 D5              PUSH   DE
2583 010816          LD     BC,1608H     ;BC = Zeiger auf Sprungtabelle
2586 09              ADD    HL,BC        ;Offset addieren
2587 4E              LD     C,(HL)       ;Adresse nach HL laden
2588 23              INC    HL
2589 66              LD     H,(HL)
258A 69              LD     L,C
258B E9              JP     (HL)         ;und Funktion ausführen
```

```
; Stringvergleich
; I: (SP-2) = RET-Adr auf 25B8H
;    (SP)   = Vektoradresse des 1. Strings
;     X     = Vektoradresse des 2. Strings
; O:  A     = FFH wenn 1.String < 2.String
;           = 00H wenn 1.String = 2.String
;           = 01H wenn 1.String > 2.String

258C CDD729     CALL  29D7H     ;2. Argument auf STR-Format
                                ;testen. HL: Vektor des
                                ;2. Strings
258F 7E         LD    A,(HL)    ;A = LEN (2.Str)
2590 23         INC   HL        ;BC = Adresse (2.Str)
2591 4E         LD    C,(HL)
2592 23         INC   HL
2593 46         LD    B,(HL)
2594 D1         POP   DE        ;Vektoradresse des 1. Strings
                                ;zurück nach DE
2595 C5         PUSH  BC        ;Adresse
2596 F5         PUSH  AF        ;und Länge des 2. Strings retten
2597 CDDE29     CALL  29DEH     ;HL: Vektor des 1. Strings
259A D1         POP   DE        ;D = Länge des 2. Strings
259B 5E         LD    E,(HL)    ;E = Länge des 1. Strings
259C 23         INC   HL        ;BC = Adresse des 1. Strings
259D 4E         LD    C,(HL)
259E 23         INC   HL
259F 46         LD    B,(HL)
25A0 E1         POP   HL        ;Adresse des 2. Strings nach HL

25A1 7B         LD    A,E       ;Beide Zähler
25A2 B2         OR    D         ;gleich Null ?
25A3 C8         RET   Z         ;Ja: Fertig
25A4 7A         LD    A,D       ;A = 2. Zähler
25A5 D601       SUB   01H       ;2. Zähler gleich Null ?
25A7 D8         RET   C         ;Ja: Fertig ( A = FFH )
25A8 AF         XOR   A         ;A = 00H
25A9 BB         CP    E         ;1. Zähler gleich Null ?
25AA 3C         INC   A         ;A = 01H wenn ja
25AB D0         RET   NC        ;Ja: Fertig ( A = 01H )
25AC 15         DEC   D         ;2. Zähler -1
25AD 1D         DEC   E         ;1. Zähler -1
25AE 0A         LD    A,(BC)    ;Nächstes Zeichen des 1. Strings
25AF BE         CP    (HL)      ;mit Zeichen des 2. Strings ver-
                                ;gleichen
25B0 23         INC   HL        ;2. Zeiger +1
25B1 03         INC   BC        ;1. Zeiger +1
25B2 28ED       JR    Z,25A1H   ;Beide identisch: Nächste Zeichen
                                ;vergleichen
25B4 3F         CCF             ;CY negieren
25B5 C36009     JP    0960H     ;A=FFH wenn CY=1 (1.Str < 2.Str)
                                ;A=01H wenn CY=0 (1.Str > 2.Str)
```

```
: Vergleichsergebnis auswerten
: I: A = FFH wenn 1. Argument < 2. Argument
:        00H wenn 1. Argument = 2. Argument
:        01H wenn 1. Argument > 2. Argument

  25B8 3C               INC    A              :A = A + 1 (CY=1 wenn A=FFH war)
  25B9 8F               ADC    A,A            :A = 01H : 1.Arg < 2.Arg
                                              :A = 02H : 1.Arg = 2.Arg
                                              :A = 04H : 1.Arg > 2.Arg
  25BA C1               POP    BC             :Vergleichsoperatorcode zurück
                                              :(siehe 2355H ff.)
  25BB A0               AND    B              :Bits der erfüllten
                                              :Operatoren = 1
  25BC C6FF             ADD    A,0FFH         :A = A + FFH, CY = 1 wenn eine
                                              :Bedingung erfüllt war (Überlauf
                                              :wenn A <> 0 war)
  25BE 9F               SBC    A,A            :A = 00H = 0 wenn nicht erfüllt
                                              :sonst A = FFH = -1
  25BF CD8D09           CALL   098DH          :X = HL = A (mit Vorzeichen)
  25C2 1812             JR     25D6H          :zurück nach 2346H

: NOT

  25C4 165A             LD     D,5AH          :Prioritätscode = 5AH (noch vor
                                              :AND und OR)
  25C6 CD3A23           CALL   233AH          :Argument nach X
  25C9 CD7F0A           CALL   0A7FH          :HL = X = CINT (X)
  25CC 7D               LD     A,L            :Ergebnis negieren
  25CD 2F               CPL
  25CE 6F               LD     L,A
  25CF 7C               LD     A,H
  25D0 2F               CPL
  25D1 67               LD     H,A
  25D2 222141           LD     (4121H),HL     :und zurückschreiben
  25D5 C1               POP    BC             :RET-Adr löschen
  25D6 C34623           JP     2346H          :und zurück nach 2346H

: RST 20H
: TSTTYP: Testet aktuellen VT und setzt Flags entsprechend
: I: -
: O: A = VT - 3
:    INT: A = FFH, Z = 0, CY = 1, S = 1
:    STR: A = 00H, Z = 1, CY = 1, S = 0
:    SNG: A = 01H, Z = 0, CY = 1, S = 0
:    DBL: A = 05H, Z = 0, CY = 0, S = 0

  25D9 3AAF40           LD     A,(40AFH)      :A = VT
  25DC FE08             CP     08H            :DBL ?
  25DE 3005             JR     NC,25E5H       :Ja: weiter bei 25E5H
  25E0 D603             SUB    03H            :Nein: A = VT - 3
  25E2 B7               OR     A              :Flags setzen
  25E3 37               SCF                   :CY = 1
  25E4 C9               RET
  25E5 D603             SUB    03H            :A = VT - 3
  25E7 B7               OR     A              :Flags setzen
  25E8 C9               RET
```

```
: Adresse suchen bzw. Variable erzeugen
: BC = 1. und 2. Zeichen des Variablennamens
: D  = Typcode der Variablen

 2652 7A          LD     A,D          :A = Typcode
 2653 32AF40      LD     (40AFH),A    :als VT ablegen
 2656 D7          RST    10H          :PTZ erhöhen
 2657 3ADC40      LD     A,(40DCH)    :Feldvariablen freigegeben ?
 265A B7          OR     A            :(siehe 1CA1H ff)
 265B C26426      JP     NZ,2664H     :Nein: Normale Variable erzeugen
 265E 7E          LD     A,(HL)       :Ja: A = nächstes Zeichen nach
                                      :Variablennamen
 265F D628        SUB    28H          :Klammer gefunden ?
 2661 CAE926      JP     Z,26E9H      :Ja: Feldvariable verarbeiten
 2664 AF          XOR    A            :A = 00
 2665 32DC40      LD     (40DCH),A    :Feldvariablen wieder freigeben
 2668 E5          PUSH   HL           :PTZ retten
 2669 D5          PUSH   DE           :Typcode retten
 266A 2AF940      LD     HL,(40F9H)   :HL = Startadresse der einfachen
                                      :Variablen
 266D EB          EX     DE,HL        :DE = HL
 266E 2AFB40      LD     HL,(40FBH)   :HL = Endadresse der einfachen
                                      :Variablen
 2671 DF          RST    18H          :Ende der einfachen Variablen
                                      :erreicht ?
 2672 E1          POP    HL           :Typcode zurück
 2673 2819        JR     Z,268EH      :Ja: Variable neu anlegen
 2675 1A          LD     A,(DE)       :A = Typcode der Variable auf die
                                      :DE zeigt
 2676 6F          LD     L,A          :Typcode nach L laden
 2677 BC          CP     H            :und mit gesuchtem Code
                                      :vergleichen
 2678 13          INC    DE           :Zeiger +1
 2679 200B        JR     NZ,2686H     :Sprung wenn Typcodes ungleich
 267B 1A          LD     A,(DE)       :2. Zeichen des Namens
 267C B9          CP     C            :vergleichen
 267D 2007        JR     NZ,2686H     :Sprung wenn ungleich
 267F 13          INC    DE           :Zeiger +1
 2680 1A          LD     A,(DE)       :1. Zeichen des Namens
 2681 B8          CP     B            :vergleichen
 2682 CACC26      JP     Z,26CCH      :Sprung wenn Variable gefunden
 2685 3E13        LD     A,13H        :--
*2686   13        INC    DE           :Zeiger +1
 2687 13          INC    DE           :Zeiger +1
 2688 E5          PUSH   HL           :Typcode retten
 2689 2600        LD     H,00H        :HL = Länge der gefundenen
                                      :Variablen
 268B 19          ADD    HL,DE        :HL auf nächste Variable erhöhen
 268C 18DF        JR     266DH        :und Vergleich wiederholen
```

```
: Variable nicht gefunden

  268E 7C           LD     A,H          ;A = Typcode
  268F E1           POP    HL           ;PTZ zurück
  2690 E3           EX     (SP),HL      ;PTZ retten, HL = RET-Adr
  2691 F5           PUSH   AF           ;Typcode retten
  2692 D5           PUSH   DE           ;Zeiger auf Ende der einfachen
                                        ;Variablen retten
  2693 11F124       LD     DE,24F1H     ;Kam der CALL auf 260DH von der
                                        ;'VARPTR'-Funktion ?
  2696 DF           RST    18H          ;Ist die RET-Adr gleich 24F1H ?
  2697 2836         JR     Z,26CFH      ;Ja: weiter bei 26CFH
  2699 114325       LD     DE,2543H     ;Nein: Kam der CALL von 2540H ?
  269C DF           RST    18H          ;Ist die RET-Adr gleich 2543H ?
  269D D1           POP    DE           ;Zeiger zurück
  269E 2835         JR     Z,26D5H      ;Ja: weiter bei 26D5H

: Neue Variable anlegen

  26A0 F1           POP    AF           ;Typcode zurück
  26A1 E3           EX     (SP),HL      ;RET-Adr retten, PTZ zurück
  26A2 E5           PUSH   HL           ;PTZ retten
  26A3 C5           PUSH   BC           ;Namen retten
  26A4 4F           LD     C,A          ;C = Typcode
  26A5 0600         LD     B,00H        ;B = 00
  26A7 C5           PUSH   BC           ;Typcode retten
                                        ;Gesamtlänge der Variablen
                                        ;errechen:
  26A8 03           INC    BC           ;BC + 1 für Typcode
  26A9 03           INC    BC           ;BC + 2 für Namen
  26AA 03           INC    BC
  26AB 2AFD40       LD     HL,(40FDH)   ;HL = Anfangsadresse des freien
                                        ;Speichers
  26AE E5           PUSH   HL           ;Adresse retten
  26AF 09           ADD    HL,BC        ;+ Gesamtlänge ergibt neue
                                        ;Anfangsadresse des freien
                                        ;Speichers
  26B0 C1           POP    BC           ;Alte Anfangsadresse zurück
  26B1 E5           PUSH   HL           ;Neue Anfangsadresse retten
  26B2 CD5519       CALL   1955H        ;Speicherplatz prüfen und
                                        ;Speicher verschieben um Platz
                                        ;für die neue Variable zu
                                        ;schaffen
  26B5 E1           POP    HL           ;Neue Anfangsadresse zurück
  26B6 22FD40       LD     (40FDH),HL   ;und abspeichern
  26B9 60           LD     H,B          ;HL = Neue Endadresse der
  26BA 69           LD     L,C          ;einfachen Variablen
  26BB 22FB40       LD     (40FBH),HL   ;Adresse abspeichern
```

```
26BE 2B            DEC     HL           :Zeiger -1
26BF 3600          LD      (HL),00H     :Speicher für neue Variable
                                        :löschen
26C1 DF            RST     18H          :Fertig ?
26C2 20FA          JR      NZ,26BEH     :Nein: weitermachen
26C4 D1            POP     DE           :Typcode zurück, HL = Adresse
                                        :der neuen Variablen
26C5 73            LD      (HL),E       :Typcode abspeichern
26C6 23            INC     HL           :Zeiger +1
26C7 D1            POP     DE           :Namen zurück nach DE
26C8 73            LD      (HL),E       :2. Zeichen abspeichern
26C9 23            INC     HL           :Zeiger +1
26CA 72            LD      (HL),D       :1. Zeichen abspeichern
26CB EB            EX      DE,HL        :DE = Zeiger auf die Mantisse der
                                        :neuen Variablen
26CC 13            INC     DE           :DE = Adresse der Variablen
26CD E1            POP     HL           :PTZ zurück
26CE C9            RET

: VARPTR-Funktion: Variable nicht gefunden

26CF 57            LD      D,A          :DE = 0000H
26D0 5F            LD      E,A
26D1 F1            POP     AF           :Stack korrigieren
26D2 F1            POP     AF
26D3 E3            EX      (SP),HL      :RET-Adr ins Stack, PTZ zurück
26D4 C9            RET                  :Fertig

: Variable bei der Ausdrucksanalyse (2337H ff) nicht gefunden
: Ergebnis auf 0 setzen

26D5 322441        LD      (4124H),A    :Exp (X) = 0 -> X = 0
26D8 C1            POP     BC           :Stack korrigieren
26D9 67            LD      H,A          :HL = 0000H
26DA 6F            LD      L,A
26DB 222141        LD      (4121H),HL   :X (INT) auf 0 setzen
26DE E7            RST     20H          :TSTTYP, String ?
26DF 2006          JR      NZ,26E7H     :Nein: X ist ok
26E1 212819        LD      HL,1928H     :Ja: Vektoradresse des Nullstrings
26E4 222141        LD      (4121H),HL   :nach X laden
26E7 E1            POP     HL           :PTZ zurück
26E8 C9            RET
```

```
: Feldvariable erkannt
: BC = Variablenname
: HL = PTZ
: A  = 00H

26E9 E5         PUSH    HL          ;PTZ retten
26EA 2AAE40     LD      HL,(40AEH)  ;L = Typcode, H = DIM-Flag
26ED E3         EX      (SP),HL     ;HL retten, PTZ zurück
26EE 57         LD      D,A         ;D = 0 (Anzahl der Dimensionen)
26EF D5         PUSH    DE          ;Register retten
26F0 C5         PUSH    BC
26F1 CD451E     CALL    1E45H       ;Nächste Dimension von (PTZ)
                                    ;holen
26F4 C1         POP     BC          ;Register zurück
26F5 F1         POP     AF
26F6 EB         EX      DE,HL       ;DE = PTZ, HL = Dimensionswert
26F7 E3         EX      (SP),HL     ;Dimensionswert retten, HL zurück
26F8 E5         PUSH    HL          ;HL wieder ins Stack setzen
26F9 EB         EX      DE,HL       ;HL = PTZ
26FA 3C         INC     A           ;Dimensionszähler +1
26FB 57         LD      D,A         ;D = Zähler
26FC 7E         LD      A,(HL)      ;Noch eine Dimension angegeben ?
26FD FE2C       CP      2CH         ;Komma als Trennung gefunden ?
26FF 28EE       JR      Z,26EFH     ;Ja: nächste Dimension holen
2701 CF         RST     08H         ;Nein: Klammer geschlossen ?
2702 29         DEFB    ')'
2703 22F340     LD      (40F3H),HL  ;PTZ abspeichern
2706 E1         POP     HL          ;Typcode und DIM-Flag zurück
2707 22AE40     LD      (40AEH),HL  ;und wieder ins RAM schreiben
270A D5         PUSH    DE          ;Dimensionszähler retten
                                    ;Im Stack stehen jetzt die
                                    ;einzelnen Dimensionswerte und
                                    ;als Abschluß die Anzahl der
                                    ;Dimensionen
270B 2AFB40     LD      HL,(40FBH)  ;HL = Zeiger auf den Anfang des
                                    ;Feldvariablenspeichers
270E 3E19       LD      A,19H       ;--
*270F   19      ADD     HL,DE       ;Feldgröße aufaddieren, HL zeigt
                                    ;jetzt auf das nächste Feld
2710 EB         EX      DE,HL
2711 2AFD40     LD      HL,(40FDH)  ;DE = Zeiger auf das Ende des
                                    ;Feldvariablenspeichers
2714 EB         EX      DE,HL
2715 DF         RST     18H         ;Ende der Feldvariablen
                                    ;erreicht ?
2716 3AAF40     LD      A,(40AFH)   ;A = Typcode des gesuchten Feldes
2719 2827       JR      Z,2742H     ;Nein: Feldvariable nicht
                                    ;gefunden, HL zeigt auf freien
                                    ;Speicherplatz für das neue Feld
```

```
 271B BE              CP     (HL)         ;Typcode gefunden ?
 271C 23              INC    HL           ;Zeiger +1
 271D 2008            JR     NZ,2727H     ;Nein: weiter bei 2727H
 271F 7E              LD     A,(HL)       ;2. Zeichen des Namens
 2720 B9              CP     C            ;vergleichen
 2721 23              INC    HL           ;Zeiger +1
 2722 2004            JR     NZ,2728H     ;Sprung wenn ungleich
 2724 7E              LD     A,(HL)       ;1. Zeichen vergleichen
 2725 B8              CP     B            ;stimmen alle Zeichen überein ?
 2726 3E23            LD     A,23H        ;--
*2727   23            INC    HL           ;Zeiger anpassen
 2728 23              INC    HL           ;Zeiger +1
 2729 5E              LD     E,(HL)       ;DE = Gesamte Feldlänge
 272A 23              INC    HL
 272B 56              LD     D,(HL)
 272C 23              INC    HL
 272D 20E0            JR     NZ,270FH     ;Feld nicht gefunden, Nächstes
                                          ;Feld überprüfen
 272F 3AAE40          LD     A,(40AEH)    ;Soll ein neues Feld angelegt
 2732 B7              OR     A            ;werden ?
 2733 1E12            LD     E,12H        ;E = Fehlercode für DD-Error
 2735 C2A219          JP     NZ,19A2H     ;Ja: Das Feld existiert bereits:
                                          ;Fehler erzeugen
 2738 F1              POP    AF           ;Nein: A = Anzahl der Dimensionen
 2739 96              SUB    (HL)         ;Stimmen auch diese überein ?
 273A CA9527          JP     Z,2795H      ;Ja: Adresse des gesuchten
                                          ;Feldelements errechnen
 273D 1E10            LD     E,10H        ;Nein: BS-Error
 273F C3A219          JP     19A2H        ;Zur Fehlerroutine springen

; Gesuchtes Feld nicht gefunden
; HL zeigt auf freien Speicherplatz für das neue Feld

 2742 77              LD     (HL),A       ;Typcode ablegen
 2743 23              INC    HL           ;Zeiger +1
 2744 5F              LD     E,A          ;DE = Länge des einzelnen Feld-
 2745 1600            LD     D,00H        ;elements
 2747 F1              POP    AF           ;A = Anzahl der Dimensionen
 2748 71              LD     (HL),C       ;Feldnamen ablegen
 2749 23              INC    HL
 274A 70              LD     (HL),B
 274B 23              INC    HL
 274C 4F              LD     C,A          ;C = Dimensionszähler
 274D CD6319          CALL   1963H        ;Noch genügend Bytes frei ?
 2750 23              INC    HL           ;Zeiger +1
 2751 23              INC    HL           ;Zeiger +1
 2752 22D840          LD     (40D8H),HL   ;Zeiger abspeichern
 2755 71              LD     (HL),C       ;Anzahl der Dimensionen ablegen
 2756 23              INC    HL           ;Zeiger +1
 2757 3AAE40          LD     A,(40AEH)    ;A = DIM-Flag
 275A 17              RLA                 ;CY = 0 wenn Adresse gesucht
                                          ;werden soll, CY = 1 wenn ein
                                          ;ganzes Feld angelegt werden soll
```

```
275B 79         LD    A,C           ;A = Dimensionszähler
275C 010B00     LD    BC,000BH      ;BC = Default-Dimensionswert
275F 3002       JR    NC,2763H      ;Sprung bei Adressensuche
2761 C1         POP   BC            ;Dimensionswert aus Stack holen
2762 03         INC   BC            ;+1 für Null-Element
2763 71         LD    (HL),C        ;Dimensionswert ablegen
2764 23         INC   HL
2765 70         LD    (HL),B
2766 23         INC   HL
2767 F5         PUSH  AF            ;Zähler retten
2768 CDAA0B     CALL  0BAAH         ;DE = DE * BC, Gesamtgröße des
                                    ;Feldes errechnen
276B F1         POP   AF            ;Zähler zurück
276C 3D         DEC   A             ;Zähler -1
276D 20ED       JR    NZ,275CH      ;Nächsten Dimensionswert
                                    ;verarbeiten
276F F5         PUSH  AF            ;00 im Stack ablegen
2770 42         LD    B,D           ;BC = Feldgröße
2771 4B         LD    C,E
2772 EB         EX    DE,HL         ;DE = Startadresse des neuen
                                    ;Feldes, HL = Feldgröße
2773 19         ADD   HL,DE         ;HL = neues Ende der Feld-
                                    ;variablen
2774 38C7       JR    C,273DH       ;BS-Error bei Speicherüberlauf
2776 CD6C19     CALL  196CH         ;Noch Speicherplatz frei ?
2779 22FD40     LD    (40FDH),HL    ;Neue Endadresse der Feld-
                                    ;variablen abspeichern
277C 2B         DEC   HL            ;Zeiger -1
277D 3600       LD    (HL),00H      ;Byte des Feldes löschen
277F DF         RST   18H           ;Feldanfang erreicht ?
2780 20FA       JR    NZ,277CH      ;Nein: nächstes Byte löschen
2782 03         INC   BC            ;BC = Anzahl der benötigten
                                    ;Bytes + 1
2783 57         LD    D,A           ;D = 0
2784 2AD840     LD    HL,(40D8H)    ;HL = Zeiger auf Anzahl der
                                    ;Dimensionen (siehe 2752H)
2787 5E         LD    E,(HL)        ;DE = Anzahl der Dimensionen
2788 EB         EX    DE,HL         ;HL = Dimensionszähler,
                                    ;DE = Zeiger
2789 29         ADD   HL,HL         ;HL = 2 * Anzahl der Dimensionen
                                    ;(Jeder Dimensionswert wird mit
                                    ;2 Bytes abgespeichert)
278A 09         ADD   HL,BC         ;+ Anzahl der für die Variablen-
                                    ;speicherung benötigten Bytes
                                    ;ergibt die Gesamtgröße des
                                    ;Feldes
278B EB         EX    DE,HL         ;DE = Gesamtlänge
278C 2B         DEC   HL            ;Zeiger -2
278D 2B         DEC   HL
278E 73         LD    (HL),E        ;Gesamtlänge abspeichern
278F 23         INC   HL
2790 72         LD    (HL),D
2791 23         INC   HL
2792 F1         POP   AF            ;Flags vom Stack holen
2793 3830       JR    C,27C5H       ;Fertig wenn nur ein neues Feld
                                    ;angelegt werden sollte (DIM)
```

: Adresse der gesuchten Variablen errechnen

```
 2795 47          LD     B,A          ;BC = 0000H
 2796 4F          LD     C,A
 2797 7E          LD     A,(HL)       ;A = Dimensionszähler
 2798 23          INC    HL           ;Zeiger +1
 2799 16E1        LD     D,0E1H       ;--
*279A   E1        POP    HL           ;Zeiger zurück
 279B 5E          LD     E,(HL)       ;DE = nächster Dimensionswert
 279C 23          INC    HL
 279D 56          LD     D,(HL)
 279E 23          INC    HL
 279F E3          EX     (SP),HL      ;Zeiger retten, HL = gesuchte
                                      ;Dimension (siehe 26F7H)
 27A0 F5          PUSH   AF           ;Zähler retten
 27A1 DF          RST    18H -        ;gesuchte Dimension gefunden ?
 27A2 D23D27      JP     NC,273DH     ;BS-Error wenn gesuchte Dimension
                                      ;> gefundene Dimension
 27A5 CDAA0B      CALL   0BAAH        ;DE = DE * BC = Gefundene
                                      ;Dimension * letzter Wert
 27A8 19          ADD    HL,DE        ;HL = Gesuchte Dimension
                                      ;+ Gefundene Dimension
                                      ;* letzter Wert
 27A9 F1          POP    AF           ;Zähler zurück
 27AA 3D          DEC    A            ;Zähler -1
 27AB 44          LD     B,H          ;BC = Feldzeiger
 27AC 4D          LD     C,L
 27AD 20EB        JR     NZ,279AH     ;Nächste Dimension verrechnen
 27AF 3AAF40      LD     A,(40AFH)    ;A = Typcode
 27B2 44          LD     B,H          ;BC = Nummer des gesuchten
 27B3 4D          LD     C,L          ;Elements
 27B4 29          ADD    HL,HL        ;* 2 ergibt Offset für INT-Feld
 27B5 D604        SUB    04H          ;Ist der Typcode INT oder STR ?
 27B7 3804        JR     C,27BDH      ;Ja: weiter bei 27BDH
 27B9 29          ADD    HL,HL        ;Nein: * 2 = Offset für SNG-Feld
 27BA 2806        JR     Z,27C2H      ;Sprung wenn SNG-Feld
 27BC 29          ADD    HL,HL        ;* 2 = Offset für DBL-Feld
 27BD B7          OR     A            ;Typcode = INT oder DBL ?
 27BE E2C227      JP     PO,27C2H     ;Ja: HL ist gesuchtes Offset
 27C1 09          ADD    HL,BC        ;Nein: Nochmals Wert aufaddieren:
                                      ;HL ist Offset für STR-Feld
 27C2 C1          POP    BC           ;Feldanfangsadresse nach BC
 27C3 09          ADD    HL,BC        ;+ Offset ergibt die gesuchte
                                      ;Adresse
 27C4 EB          EX     DE,HL        ;DE = Adresse
 27C5 2AF340      LD     HL,(40F3H)   ;HL = PTZ
 27C8 C9          RET
```

```
: MEM = FRE ( Numerische Variable )

   27C9 AF              XOR     A               ;A = 0
   27CA E5              PUSH    HL              ;PTZ retten
   27CB 32AF40          LD      (40AFH),A       ;Typcode auf 0 setzen
   27CE CDD427          CALL    27D4H           ;X = Ende des Stacks - Beginn des
                                                ;freien Speichers = Anzahl der
                                                ;freien Bytes
   27D1 E1              POP     HL              ;PTZ zurück
   27D2 D7              RST     10H             ;PTZ erhöhen
   27D3 C9              RET

: FRE

   27D4 2AFD40          LD      HL,(40FDH)      ;HL = Beginn des freien Speichers
                                                ;(= Ende der Feldvariablen)
   27D7 EB              EX      DE,HL           ;DE = HL
   27D8 210000          LD      HL,0000H        ;HL = 0
   27DB 39              ADD     HL,SP           ;HL = 0 + SP = SP
   27DC E7              RST     20H             ;TSTTYP, War das Argument
                                                ;im STR-Format ?
                                                ;(wird die Anzahl der freien
                                                ;Bytes im Stringspeicher
                                                ;gewünscht ?)
   27DD 200D            JR      NZ,27ECH        ;Nein: weiter bei 27ECH
   27DF CDDA29          CALL    29DAH           ;Ja: Argument aus dem String-
                                                ;speicher wieder löschen
   27E2 CDE628          CALL    28E6H           ;Stringspeicher sortieren
   27E5 2AA040          LD      HL,(40A0H)      ;HL = Start des Stringspeichers
   27E8 EB              EX      DE,HL           ;DE = HL
   27E9 2AD640          LD      HL,(40D6H)      ;HL = Adresse des letzten Strings
                                                ;im Stringspeicher
   27EC 7D              LD      A,L             ;Beide Werte subtrahieren
   27ED 93              SUB     E
   27EE 6F              LD      L,A
   27EF 7C              LD      A,H
   27F0 9A              SBC     A,D
   27F1 67              LD      H,A
   27F2 C3660C          JP      0C66H           ;HL als Ergebnis in X ablegen

: POS

   27F5 3AA640          LD      A,(40A6H)       ;A = aktuelle Cursorposition
   27F8 6F              LD      L,A             ;HL = A (ohne Vorzeichen)
   27F9 AF              XOR     A
   27FA 67              LD      H,A
   27FB C39A0A          JP      0A9AH           ;X = HL (INT)
```

```
; USR

  27FE CDA941          CALL    41A9H          ;DOS
  2801 D7              RST     10H            ;PTZ erhöhen
  2802 CD2C25          CALL    252CH          ;Klammerausdruck bearbeiten
  2805 E5              PUSH    HL             ;PTZ retten
  2806 219008          LD      HL,0890H       ;RET-Adr auf 0890H setzen
  2809 E5              PUSH    HL
  280A 3AAF40          LD      A,(40AFH)      ;A = Typcode
  280D F5              PUSH    AF             ;Typcode retten
  280E FE03            CP      03H            ;Stringargument ?
  2810 CCDA29          CALL    Z,29DAH        ;Ja: Argument aus Stringspeicher
                                              ;löschen
  2813 F1              POP     AF             ;Typcode zurück
  2814 EB              EX      DE,HL          ;DE = 0890H (bei numerischem
                                              ;Argument)
                                              ;DE = Vektoradresse bei String-
                                              ;argument
  2815 2A8E40          LD      HL,(408EH)     ;HL = Adresse der USR-Routine
  2818 E9              JP      (HL)           ;Routine anspringen

; Umwandlung von X in gewünschten Typ
; I: A = gewünschter Typcode
; (siehe 1F35H ff)

  2819 E5              PUSH    HL             ;HL retten
  281A E607            AND     07H            ;A = Offset für Sprungtabelle:
                                              ;INT: A = 2, STR: A = 3
                                              ;SNG: A = 4, DBL: A = 0
  281C 21A118          LD      HL,18A1H       ;HL = Sprungtabelle für
                                              ;Typumwandlung
  281F 4F              LD      C,A            ;BC = Offset
  2820 0600            LD      B,00H
  2822 09              ADD     HL,BC          ;Offset aufaddieren
  2823 CD8625          CALL    2586H          ;Nochmals Offset aufaddieren
                                              ;(2 Bytes Adresse) und Routine
                                              ;ausführen
  2826 E1              POP     HL             ;HL zurück
  2827 C9              RET

; UPRO für INPUT
; Test auf ID-Error
; (siehe 219AH)

  2828 E5              PUSH    HL             ;PTZ retten
  2829 2AA240          LD      HL,(40A2H)     ;HL = aktuelle ZN
  282C 23              INC     HL             ;Aktuelle ZN = 65535 ?
  282D 7C              LD      A,H            ;(HL + 1 = 0000H ?)
  282E B5              OR      L
  282F E1              POP     HL             ;PTZ zurück
  2830 C0              RET     NZ             ;Nein: OK
```

```
; ID-Error

  2831 1E16            LD      E,16H        ;Ja: ID-Error
  2833 C3A219          JP      19A2H

; STR$

  2836 CDBD0F          CALL    0FBDH        ;X in String ab 4130H umwandeln
  2839 CD6528          CALL    2865H        ;Stringkonstante ab (HL)
                                            ;übernehmen
  283C CDDA29          CALL    29DAH        ;und wieder aus dem String-
                                            ;speicher entfernen
                                            ;HL: Vektor des Strings
  283F 012B2A          LD      BC,2A2BH     ;RET-Adr auf 2A2BH setzen
  2842 C5              PUSH    BC

; Neuen String in den Stringspeicher übernehmen
; I: HL: Vektor des neuen Strings (irgendwo im Speicher)
; O: DE: Vektor des neuen Strings (im Stringspeicher)

  2843 7E              LD      A,(HL)       ;A = Stringlänge
  2844 23              INC     HL           ;Zeiger +1
  2845 E5              PUSH    HL           ;Zeiger retten
  2846 CDBF28          CALL    28BFH        ;Sind noch A Bytes Platz im
                                            ;Stringspeicher ?
                                            ;Ja: DE = Zeiger auf 1. freies
                                            ;Byte für den neuen String
                                            ;Nein: OS-Error
  2849 E1              POP     HL           ;Zeiger zurück
  284A 4E              LD      C,(HL)       ;Stringadresse nach BC holen
  284B 23              INC     HL
  284C 46              LD      B,(HL)
  284D CD5A28          CALL    285AH        ;Länge und Adresse des freien
                                            ;Speichers im Stringspeicher in
                                            ;der Stringtabelle als letzten
                                            ;Eintrag ablegen
  2850 E5              PUSH    HL           ;Vektoradresse des neuen String-
                                            ;platzes retten
  2851 6F              LD      L,A          ;L = Länge des Strings
  2852 CDCE29          CALL    29CEH        ;String von (BC) nach (DE)
                                            ;kopieren (String in String-
                                            ;speicher übernehmen)
  2855 D1              POP     DE           ;DE: Vektor des neuen Strings
  2856 C9              RET
```

```
; Platz im Stringspeicher suchen
; I: A  = Länge des neuen Strings
; O: DE = Startadresse im Stringspeicher für den neuen String
;    HL = 40D3H = Vektoradresse des freien Strings

  2857 CDBF28         CALL   28BFH        ;Noch A Bytes Platz im String-
                                          ;speicher ?
  285A 21D340         LD     HL,40D3H     ;HL: Vektor des freien Strings
  285D E5             PUSH   HL           ;Vektoradresse retten
  285E 77             LD     (HL),A       ;Länge ablegen
  285F 23             INC    HL
  2860 73             LD     (HL),E       ;Adresse des freien Speichers
  2861 23             INC    HL           ;im Stringspeicher ablegen
  2862 72             LD     (HL),D
  2863 E1             POP    HL           ;Vektoradresse zurück
  2864 C9             RET      -

; Stringkonstante bei (HL) aufarbeiten und in Stringtabelle übernehmen
; I: HL zeigt auf Stringkonstante (mit '"' oder 00H abgeschlossen)
; O: A  = letztes Zeichen des Strings
;    HL = Zeiger auf Stringende
;    X  = Vektoradresse des neuen Strings

  2865 2B             DEC    HL           ;Zeiger -1
  2866 0622           LD     B,22H        ;B = 1. Endmarke ('"')
  2868 50             LD     D,B          ;D = 2. Endmarke
  2869 E5             PUSH   HL           ;Zeiger retten
  286A 0EFF           LD     C,0FFH       ;Längenzähler auf -1 setzen
  286C 23             INC    HL           ;Zeiger +1
  286D 7E             LD     A,(HL)       ;Zeichen holen
  286E 0C             INC    C            ;Zähler +1
  286F B7             OR     A            ;Stringende (00H) erreicht ?
  2870 2806           JR     Z,2878H      ;Ja: weiter bei 2878H
  2872 BA             CP     D            ;2. Marke erreicht ?
  2873 2803           JR     Z,2878H      ;Ja: String zuende
  2875 B8             CP     B            ;Startmarke erreicht ('"')
  2876 20F4           JR     NZ,286CH     ;Nein: nächstes Zeichen holen
  2878 FE22           CP     22H          ;Wurde der String durch '"'
                                          ;beendet ?
  287A CC781D         CALL   Z,1D78H      ;Ja: Zeiger erhöhen
  287D E3             EX     (SP),HL      ;Endzeiger retten,
                                          ;HL = Startzeiger
  287E 23             INC    HL           ;Zeiger +1
  287F EB             EX     DE,HL        ;DE = Startzeiger
  2880 79             LD     A,C          ;C = Länge
  2881 CD5A28         CALL   285AH        ;Länge und Zeiger als letzten
                                          ;Eintrag in die Stringtabelle
                                          ;übernehmen
```

```
  2884 11D340             LD      DE,40D3H       :DE: Vektor des Strings
  2887 3ED5               LD      A,0D5H         :--
  2889 2AB340             LD      HL,(40B3H)     :HL = nächste freie Position in
                                                 :der Stringtabelle
  288C 222141             LD      (4121H),HL     :Vektoradresse in X ablegen
  288F 3E03               LD      A,03H          :VT auf String setzen
  2891 32AF40             LD      (40AFH),A
  2894 CDD309             CALL    09D3H          :3 Bytes von (DE) nach (HL)
                                                 :kopieren (Länge und Adresse in
                                                 :die Stringtabelle übernehmen)
  2897 11D640             LD      DE,40D6H       :Wurde das Ende der Stringtabelle
  289A DF                 RST     18H            :erreicht ? (Ist HL = 40D6H ?)
  289B 22B340             LD      (40B3H),HL     :Tabellenzeiger abspeichern
  289E E1                 POP     HL             :Endzeiger zurück
  289F 7E                 LD      A,(HL)         :A = letztes Zeichen
  28A0 C0                 RET     NZ -           :Nein: Fertig

; ST-Error

  28A1 1E1E               LD      E,1EH          :Ja: ST-Error
  28A3 C3A219             JP      19A2H

; Textkonstante ab (HL) aufarbeiten und ausgeben
; I: HL zeigt auf eine Stringkonstante

  28A6 23                 INC     HL             :Zeiger +1
  28A7 CD6528             CALL    2865H          :Stringkonstante übernehmen
  28AA CDDA29             CALL    29DAH          :und wieder aus Stringtabelle
                                                 :löschen
  28AD CDC409             CALL    09C4H          :D = Länge, BC = Startadresse des
                                                 :Strings
  28B0 14                 INC     D              :Zähler +1
  28B1 15                 DEC     D              :Zähler -1, Zähler = 0 ?
  28B2 C8                 RET     Z              :Ja: Fertig
  28B3 0A                 LD      A,(BC)         :Zeichen holen
  28B4 CD2A03             CALL    032AH          :und ausgeben
  28B7 FE0D               CP      0DH            :War es ein CR/LF ?
  28B9 CC0321             CALL    Z,2103H        :Ja: DOS anspringen
  28BC 03                 INC     BC             :Zeiger +1
  28BD 18F2               JR      28B1H          :nächstes Zeichen holen

; Platz im Stringspeicher prüfen
; Falls weniger als A Bytes im Stringspeicher frei sind wird der Stringspeicher
; umsortiert und die Routine nocheinmal ausgeführt.
; Ist der Stringspeicher voll, dann wird ein OS-Error erzeugt
;
; I: A  = Anzahl der im Stringspeicher benötigten Bytes
; O: DE = Zeiger auf A Bytes freien Platz im Stringspeicher
;    HL = Startadresse des Stringspeichers

  28BF B7                 OR      A              :Z = 0 als Flag für den ersten
                                                 :Durchlauf
  28C0 0EF1               LD      C,0F1H         :--
```

```
: Ansprung beim zweiten Durchlauf

*28C1   F1          POP    AF          :AF zurück (jetzt ist Z = 1 !)
 28C2 F5            PUSH   AF          :AF retten
 28C3 2AA040        LD     HL,(40A0H)  :HL = Startadresse des
                                       :Stringspeichers
 28C6 EB            EX     DE,HL       :DE = HL
 28C7 2AD640        LD     HL,(40D6H)  :HL = Adresse des ersten freien
                                       :Bytes im Stringspeicher (der
                                       :Stringspeicher wird von OBEN
                                       :nach UNTEN gefüllt !)
 28CA 2F            CPL                :A = -A
 28CB 4F            LD     C,A         :BC = - Anzahl der benötigten
 28CC 06FF          LD     B,0FFH      :Bytes
 28CE 09            ADD    HL,BC       :HL + (-Anzahl der benötigten
                              -        :Bytes) = Startadresse für den
                                       :neuen String
 28CF 23            INC    HL          :+1 da HL bereits auf ein freies
                                       :Byte zeigte
 28D0 DF            RST    18H         :Anfang des Stringspeichers
                                       :unterschritten ?
 28D1 3807          JR     C,28DAH     :Ja: Stringspeicher sortieren
                                       :OS-Error beim 2. Durchlauf
 28D3 22D640        LD     (40D6H),HL  :Nein: Zeiger abspeichern
 28D6 23            INC    HL          :Zeiger +1
 28D7 EB            EX     DE,HL       :DE = Zeiger auf freien
                                       :Speicherplatz
 28D8 F1            POP    AF          :AF zurück
 28D9 C9            RET

; Stringspeicher sortieren
; Es kann sein, daß noch Strings im Stringspeicher stehen, deren Variablen
; schon gelöscht sind. Diese Routine prüft dies nach und verschiebt den
; Stringspeicher entsprechend
; I: (SP) = Flag für 1. Durchlauf

 28DA F1            POP    AF          :Flag zurück
                                       :Ist dies schon der zweite
                                       :Durchlauf ?
 28DB 1E1A          LD     E,1AH       :E = Fehlercode für OS-Error
 28DD CAA219        JP     Z,19A2H     :Ja: Fehler erzeugen
 28E0 BF            CP     A           :Z = 1 setzen (Kennung für
                                       :zweiten Durchlauf)
 28E1 F5            PUSH   AF          :Flag retten
 28E2 01C128        LD     BC,28C1H    :RET-Adr auf 28C1H setzen
 28E5 C5            PUSH   BC          :(Für 2. Durchlauf)
 28E6 2AB140        LD     HL,(40B1H)  :HL = Endadresse des String-
                                       :speichers
                                       := Adresse des höchsten Strings
                                       :im Stringspeicher (Default)
```

```
; Nächsten höchsten String im Stringspeicher suchen und einsortieren
; (Höchster String heißt der String mit der größten Startadresse)
; HL = Adresse des zuletzt einsortierten Strings im Stringspeicher
; (= Adresse des letzten Strings im Stringspeicher beim letzten Durchlauf)

  28E9 22D640          LD      (40D6H),HL    ;Adresse des zuletzt einsor-
                                             ;tierten Strings abspeichern
  28EC 210000          LD      HL,0000H      ;HL = 0000H (= Defaultadresse)
  28EF E5              PUSH    HL            ;0000H ins Stack schreiben
  28F0 2AA040          LD      HL,(40A0H)    ;HL = Startadresse des
                                             ;Stringspeichers
  28F3 E5              PUSH    HL            ;Startadresse retten

; Bei der Startadresse wird die Prüfung auf höchsten String begonnen und endet
; bei der Startadresse des zuletzt einsortierten Strings.
; Der in diesem Bereich höchste String wird dann neu einsortiert und die
; Routine solange wiederholt, bis kein höchster String mehr gefunden wird.
; (d.h. alle Strings einsortiert sind)
;
; Suche des nächsten höchsten Strings beginnen

  28F4 21B540          LD      HL,40B5H      ;HL = Startadresse der String-
                                             ;tabelle = Vektoradresse des
                                             ;ersten Strings in der String-
                                             ;tabelle

; Alle Stringvariablen überprüfen
; Es wird zuerst bei den Einträgen in der Stringtabelle, dann bei den einfachen
; Stringvariablen und zum Schluß bei den Feldvariablen überprüft, ob sie im
; Stringspeicher stehen

  28F7 EB              EX      DE,HL         ;DE = Adresse des nächsten freien
  28F8 2AB340          LD      HL,(40B3H)    ;Eintrags in der Stringtabellen
  28FB EB              EX      DE,HL
  28FC DF              RST     18H           ;Ende der Tabelle erreicht ?
  28FD 01F728          LD      BC,28F7H      ;BC = RET-Adr
  2900 C24A29          JP      NZ,294AH      ;Nein: weiter bei 294AH

; Ende der Stringtabelle erreicht
; Einfache Stringvariablen überprüfen

  2903 2AF940          LD      HL,(40F9H)    ;HL = Startadresse der einfachen
                                             ;Variablen
  2906 EB              EX      DE,HL
  2907 2AFB40          LD      HL,(40FBH)    ;DE = Startadresse der Feld-
  290A EB              EX      DE,HL         ;variablen
  290B DF              RST     18H           ;Ende der einfachen Variablen
                                             ;erreicht ?
  290C 2813            JR      Z,2921H       ;Ja: weiter bei 2921H
```

```
290E 7E         LD      A,(HL)       :A = Typcode
290F 23         INC     HL           :Zeiger +1 (für Typcode)
2910 23         INC     HL           :Zeiger +2 (für Namen)
2911 23         INC     HL
2912 FE03       CP      03H          :Stringvariable gefunden ?
2914 2004       JR      NZ,291AH     :Nein: nächste Variable prüfen
2916 CD4B29     CALL    294BH        :Ja: Adresse überprüfen
2919 AF         XOR     A            :A = 00H
291A 5F         LD      E,A          :DE = Offset zur nächsten
291B 1600       LD      D,00H        :Variablen
291D 19         ADD     HL,DE        :Offset addieren:
                                     :HL: Vektor der nächsten
                                     :Variablen
291E 18E6       JR      2906H        :nächste Variable prüfen

; Ende der einfachen Variablen erreicht
; Feldvariablen überprüfen

2920 C1         POP     BC           :Stackkorrektur
2921 EB         EX      DE,HL
2922 2AFD40     LD      HL,(40FDH)   :DE = Endadresse der Feld-
2925 EB         EX      DE,HL        :variablen
2926 DF         RST     18H          :Ende der Feldvariablen
                                     :erreicht ?
2927 CA6B29     JP      Z,296BH      :Ja: Höchsten String einsortieren
292A 7E         LD      A,(HL)       :A = Typcode des Feldes
292B 23         INC     HL           :Zeiger +1
292C CDC209     CALL    09C2H        :BCDE = (HL): DE = Namen,
                                     :BC = Länge des Feldes
292F E5         PUSH    HL           :Zeiger retten
2930 09         ADD     HL,BC        :HL = Zeiger auf nächstes Feld
2931 FE03       CP      03H          :Stringfeld gefunden ?
2933 20EB       JR      NZ,2920H     :Nein: nächstes Feld testen
2935 22D840     LD      (40D8H),HL   :Ja: Zeiger retten
2938 E1         POP     HL           :HL = Zeiger auf die Anzahl der
                                     :Dimensionen
2939 4E         LD      C,(HL)       :BC = Anzahl der Dimensionen
293A 0600       LD      B,00H
293C 09         ADD     HL,BC        :Anzahl zweimal aufaddieren
                                     :(Jede Dimension wurde mit 2
                                     :Bytes abgespeichert)
293D 09         ADD     HL,BC        :HL ist jetzt Vektoradresse des
                                     :ersten Feldelements -1
293E 23         INC     HL           :HL +1: Vektor
293F EB         EX      DE,HL        :DE = Zeiger auf das nächste Feld
2940 2AD840     LD      HL,(40D8H)   :(= Endadresse des aktuellen
2943 EB         EX      DE,HL        :Feldes, siehe 2935H)
2944 DF         RST     18H          :Ende des Feldes erreicht ?
2945 28DA       JR      Z,2921H      :Ja: nächstes Feld überprüfen
2947 013F29     LD      BC,293FH     :Nein: RET-Adr auf 293FH setzen
294A C5         PUSH    BC
```

```
; Stringadresse der gefundenen Stringvariablen überprüfen
; Dazu zwei Kriterien:
; 1. Steht der String im Stringspeicher und ist noch nicht wieder übernommen
;    worden ?
;    (d.h. Ist die Stringadresse kleiner als die des zuletzt einsortierten
;    Strings)
; 2. Steht der String höher im Stringspeicher als der zuletzt als höchster
;    String angenommene String ?
;    (d.h. Ist die Stringadresse größer als die. des zuletzt als höchsten
;    String angenommenen Strings ?)
;
; Wenn beide Kriterien zutreffen, wird die Adresse des gefundenen Strings im
; Stack als größte Adresse abgelegt.
;
; I: HL = Vektoradresse der Stringvariablen
; O: HL = Vektoradresse auf die nächste Variable

 294B AF          XOR   A           ;A = 00H
 294C B6          OR    (HL)        ;A = Länge des Strings
 294D 23          INC   HL          ;Zeiger +1
 294E 5E          LD    E,(HL)      ;DE = Stringadresse
 294F 23          INC   HL
 2950 56          LD    D,(HL)
 2951 23          INC   HL
 2952 C8          RET   Z           ;Fertig wenn ein String mit der
                                    ;Länge Null gefunden wurde

; 1. Kriterium

 2953 44          LD    B,H         ;BC: Vektor der nächsten
 2954 4D          LD    C,L         ;Variablen
 2955 2AD640      LD    HL,(40D6H)  ;HL = Adresse des zuletzt ein-
                                    ;sortierten Strings
 2958 DF          RST   18H         ;Ist der gefundene String schon
                                    ;wieder in den Stringspeicher
                                    ;übernommen worden ?
                                    ;(Ist die Stringadresse größer
                                    ;als die Adresse des zuletzt ein-
                                    ;sortierten Strings ?)
 2959 60          LD    H,B         ;HL: Vektor
 295A 69          LD    L,C
 295B D8          RET   C           ;Ja: Fertig

; 2. Kriterium

 295C E1          POP   HL          ;Nein: HL = RET-Adr
 295D E3          EX    (SP),HL     ;RET-Adr retten
                                    ;HL = Adresse des zuletzt als
                                    ;höchsten String angenommen
                                    ;Strings (siehe 2968H ff)
 295E DF          RST   18H         ;Ist die Startadresse des Strings
                                    ;größer als HL ?
 295F E3          EX    (SP),HL     ;HL und
 2960 E5          PUSH  HL          ;RET-Adr ins Stack zurück
 2961 60          LD    H,B         ;HL: Vektor
 2962 69          LD    L,C
 2963 D0          RET   NC          ;Nein: Fertig
```

```
; Neuen höchsten String gefunden
; Stringadresse und Vektoradresse ins Stack retten

  2964 C1          POP    BC           :BC = RET-Adr
  2965 F1          POP    AF           :Startadresse des zuletzt als
                                       :höchsten String angenommenen
                                       :Strings
  2966 F1          POP    AF           :und dessen Vektoradresse
                                       :vom Stack entfernen
  2967 E5          PUSH   HL           :Dafür wird die jetzige Vektor-
                                       :adresse
  2968 D5          PUSH   DE           :und die Startadresse ins Stack
                                       :übernommen
  2969 C5          PUSH   BC           :RET-Adr zurück ins Stack
  296A C9          RET

; Alle Stringvariablen überprüft
; Neuen höchsten String an den zuletzt einsortierten String anhängen
; und damit allen dazwischenliegenden 'Müll' löschen

  296B D1          POP    DE           :Adresse des höchsten Strings vom
                                       :Stack nehmen
  296C E1          POP    HL           :Vektoradresse zurück
  296D 7D          LD     A,L          :Wurde die Adresse seit 28ECH
                                       :geändert ?
  296E B4          OR     H            :(d.h. wurde ein neuer höchster
                                       :String gefunden ?)
  296F C8          RET    Z            :Nein: Fertig

; Es wurde ein neuer höchster String gefunden

  2970 2B          DEC    HL           :BC = Adresse des neuen höchsten
  2971 46          LD     B,(HL)       :Strings
  2972 2B          DEC    HL           :(= DE, da die Adresse zu DE ge-
  2973 4E          LD     C,(HL)       :hört)
  2974 E5          PUSH   HL           :Adresse +1 retten
  2975 2B          DEC    HL           :HL -1 = Vektoradresse des
                                       :zuletzt überprüften Strings
  2976 6E          LD     L,(HL)       :HL = Stringlänge
  2977 2600        LD     H,00H
  2979 09          ADD    HL,BC        :HL = Länge + Stringadresse
  297A 50          LD     D,B          :DE = Stringadresse
  297B 59          LD     E,C
  297C 2B          DEC    HL           :HL = Endadresse des Strings
  297D 44          LD     B,H          :BC = HL
  297E 4D          LD     C,L
  297F 2AD640      LD     HL,(40D6H)   :HL = Adresse des zuletzt einsor-
                                       :tierten Strings
  2982 CD5819      CALL   1958H        :Durch Anhängen des neuen Strings
                                       :an den zuletzt einsortierten,
                                       :den neuen String einsortieren
```

```
 2985 E1             POP    HL           ;Vektoradresse +1 zurück
 2986 71             LD     (HL),C       ;neue Startadresse einsetzen
 2987 23             INC    HL
 2988 70             LD     (HL),B
 2989 69             LD     L,C          ;HL = Startadresse des über-
 298A 60             LD     H,B          ;nommenen Strings
 298B 2B             DEC    HL           ;HL -1 = Adresse des zuletzt ein-
                                         ;sortierten Strings
 298C C3E928         JP     28E9H        ;nächsten höchsten String suchen

; Stringaddition
; I: HL = PTZ auf '+'
;    BC = Prioritätscode
;    X  = 1. String
; O: X  = neuer String

 298F C5             PUSH   BC           ;Prioritätscode retten
 2990 E5             PUSH   HL           ;PTZ retten
 2991 2A2141         LD     HL,(4121H)   ;HL: Vektor des 1. Strings
 2994 E3             EX     (SP),HL      ;Vektoradresse retten, PTZ zurück
 2995 CD9F24         CALL   249FH        ;2. String nach X holen
 2998 E3             EX     (SP),HL      ;PTZ retten, Vektoradresse des
                                         ;1. Strings zurück
 2999 CDF40A         CALL   0AF4H        ;TM-Error wenn in X kein String
                                         ;(2. Argument falsch)
 299C 7E             LD     A,(HL)       ;A = Länge des 1. Strings
 299D E5             PUSH   HL           ;Vektoradresse 1. String retten
 299E 2A2141         LD     HL,(4121H)   ;HL: Vektor des 2. Strings
 29A1 E5             PUSH   HL           ;Vektoradresse 2. String retten
 29A2 86             ADD    A,(HL)       ;Länge des 2. Strings zu A
                                         ;addieren: A = Gesamtlänge des
                                         ;neuen Strings
 29A3 1E1C           LD     E,1CH        ;E = Fehlercode für LS-Error
 29A5 DAA219         JP     C,19A2H      ;Fehler bei Überlauf (neuer
                                         ;String ist zu lang)
 29A8 CD5728         CALL   2857H        ;A Bytes Platz im Stringspeicher
                                         ;schaffen, HL: Vektor des
                                         ;neuen String(platzes)
 29AB D1             POP    DE           ;Vektoradresse 2. String zurück
 29AC CDDE29         CALL   29DEH        ;2. String aus Tabelle und
                                         ;Stringspeicher löschen ?
 29AF E3             EX     (SP),HL      ;Vektoradresse 2. String retten,
                                         ;Vektoradresse 1. String zurück
 29B0 CDDD29         CALL   29DDH        ;1. String aus Tabelle und
                                         ;Stringspeicher löschen ?
 29B3 E5             PUSH   HL           ;Vektoradresse 1. String retten
 29B4 2AD440         LD     HL,(40D4H)   ;HL = Adresse des freien Platzes
                                         ;für den neuen String
 29B7 EB             EX     DE,HL        ;DE = HL
```

```
  29B8 CDC629          CALL   29C6H        ;1. String in freien Platz
                                           ;kopieren
  29BB CDC629          CALL   29C6H        ;2. String dahinter kopieren
  29BE 214923          LD     HL,2349H     ;Bei 2349H wieder in die
                                           ;Ausdrucksdecodierung
                                           ;zurückspringen
  29C1 E3              EX     (SP),HL      ;RET-Adr ins Stack
  29C2 E5              PUSH   HL           ;Alte RET-Adr wieder zurück-
                                           ;schreiben
  29C3 C38428          JP     2884H        ;Neuen String in Stringtabelle
                                           ;übernehmen

; String nach (DE) kopieren
; I: (SP - 2) : Vektor des Strings
;        DE   = Neue Adresse des Strings

  29C6 E1              POP    HL           ;RET-Adr nach HL
  29C7 E3              EX     (SP),HL      ;Vektoradresse holen,
                                           ;RET-Adr ins Stack zurück
  29C8 7E              LD     A,(HL)       ;A = Länge
  29C9 23              INC    HL
  29CA 4E              LD     C,(HL)       ;BC = Adresse des Strings
  29CB 23              INC    HL
  29CC 46              LD     B,(HL)
  29CD 6F              LD     L,A          ;L = Zähler
  29CE 2C              INC    L            ;Zähler = 0 ?
  29CF 2D              DEC    L            ;Zähler -1
  29D0 C8              RET    Z            ;Ja: Fertig
  29D1 0A              LD     A,(BC)       ;Nein: Zeichen von (BC)
  29D2 12              LD     (DE),A       ;nach (DE) kopieren
  29D3 03              INC    BC           ;Zeiger +1
  29D4 13              INC    DE           ;Zeiger +1
  29D5 18F8            JR     29CFH        ;Nächstes Byte

; String in X aus Tabelle und Stringspeicher entfernen ?
;
; Bei Stringfunktionen werden grundsätzlich alle Stringkonstanten bzw.
; Zwischenergebnisse die als Argumente auftreten in die Stringtabelle
; und in den Stringspeicher übernommen.
; Da diese Strings aber nur von der Funktion gebraucht werden, können sie
; wieder gelöscht werden.

; I: X  = String
; O: HL : Vektor des Strings

  29D7 CDF40A          CALL   0AF4H        ;Test ob X im STR-Format
  29DA 2A2141          LD     HL,(4121H)   ;HL : Vektor des Strings
```

```
: String aus Tabelle und Stringspeicher löschen ?
: I: HL : Vektor des Strings

  29DD EB            EX     DE,HL        ;DE : Vektor des Strings
  29DE CDF529        CALL   29F5H        ;Zeigt die Vektoradresse auf den
                                         ;letzten Eintrag in der String-
                                         ;tabelle, wird dieser gelöscht
                                         ;(In diesem Fall handelt es sich
                                         ;um ein Zwischenergebnis oder um
                                         ;eine Stringkonstante !)
  29E1 EB            EX     DE,HL        ;HL : Vektor
  29E2 C0            RET    NZ           ;Fertig wenn der letzte Eintrag
                                         ;nicht gelöscht wurde

  29E3 D5            PUSH   DE           ;Vektoradresse retten
  29E4 50            LD     D,B          ;DE = Stringadresse
  29E5 59            LD     E,C
  29E6 1B            DEC    DE           ;DE -1
  29E7 4E            LD     C,(HL)       ;C = Länge des Strings
  29E8 2AD640        LD     HL,(40D6H)   ;HL = Adresse des letzten Strings
                                         ;im Stringspeicher -1
  29EB DF            RST    18H          ;Ist der String der letzte im
                                         ;Stringspeicher ?
                                         ;(In diesem Fall handelt es sich
                                         ;um eine Stringkonstante !)
  29EC 2005          JR     NZ,29F3H     ;Nein: Fertig

  29EE 47            LD     B,A          ;Ja: B = 0, BC = Stringlänge
  29EF 09            ADD    HL,BC        ;Stringlänge zu HL addieren:
                                         ;HL zeigt jetzt auf den neuen
                                         ;letzten String im Stringspeicher
  29F0 22D640        LD     (40D6H),HL   ;Neue Adresse zurückschreiben
                                         ;(die Stringkonstante ist jetzt
                                         ;aus dem Stringspeicher gelöscht)
  29F3 E1            POP    HL           ;Vektoradresse zurück
  29F4 C9            RET

: Zeigt DE auf den letzten Eintrag der Stringtabelle ?
: Wenn ja, dann wird der letzte Eintrag gelöscht
: I: DE : Vektor des Strings
: O: BC = Adresse des Strings im letzten Eintrag der Stringtabelle
:         (= Stringadresse wenn Z = 1)
:    DE : Vektor des Strings
:    HL = DE
:    Z = 1: der Eintrag wurde gelöscht

  29F5 2AB340        LD     HL,(40B3H)   ;HL = Adresse des nächsten freien
                                         ;Eintrags in der Stringtabelle
  29F8 2B            DEC    HL
  29F9 46            LD     B,(HL)       ;BC = Adresse des Strings
  29FA 2B            DEC    HL
  29FB 4E            LD     C,(HL)
  29FC 2B            DEC    HL
  29FD DF            RST    18H          ;Zeigt DE auf diesen Eintrag ?
  29FE C0            RET    NZ           ;Nein: Fertig
  29FF 22B340        LD     (40B3H),HL   ;Ja: Neue Adresse zurückschreiben
                                         ;(alter Eintrag wird über-
                                         ;schrieben)
  2A02 C9            RET
```

```
: LEN

  2A03 01F827           LD      BC.27F8H       ;RET-Adr auf 27F8H setzen
  2A06 C5               PUSH    BC
  2A07 CDD729           CALL    29D7H          ;X auf Stringformat prüfen
                                               ;String aus Tabelle und String-
                                               ;speicher löschen ?
                                               ;HL : Vektor des Stringarguments
  2A0A AF               XOR     A              ;A = 00
  2A0B 57               LD      D.A            ;D = 00
  2A0C 7E               LD      A.(HL)         ;A = Stringlänge
  2A0D B7               OR      A              ;Flags setzen
  2A0E C9               RET                    ;RET nach 27F8H: A als INT nach X
                                               ;schreiben

: ASC

  2A0F 01F827           LD      BC.27F8H       ;RET-Adr auf 27F8H setzen
  2A12 C5               PUSH    BC
  2A13 CD072A           CALL    2A07H          ;Vektoradresse und Länge des
                                               ;Stringarguments holen
  2A16 CA4A1E           JP      Z.1E4AH        ;FC-Error wenn Nullstring
  2A19 23               INC     HL             ;Adresse nach DE holen
  2A1A 5E               LD      E.(HL)
  2A1B 23               INC     HL
  2A1C 56               LD      D.(HL)
  2A1D 1A               LD      A.(DE)         ;A = 1. Zeichen des Strings
  2A1E C9               RET                    ;RET nach 27F8H (siehe 2A0EH)

: CHR$

  2A1F 3E01             LD      A.01H          ;A = Länge des Ergebnisstrings
  2A21 CD5728           CALL    2857H          ;Platz im Stringspeicher schaffen
  2A24 CD1F2B           CALL    2B1FH          ;Argument nach DE
  2A27 2AD440           LD      HL.(40D4H)     ;HL = Stringadresse
  2A2A 73               LD      (HL).E         ;Wert in String schreiben
  2A2B C1               POP     BC             ;RET-Adr löschen
  2A2C C38428           JP      2884H          ;String in Speicher übernehmen

: STRING$

  2A2F D7               RST     10H            ;PTZ erhöhen
  2A30 CF               RST     08H            ;Klammer auf ?
  2A31 28               DEFB    '('
  2A32 CD1C2B           CALL    2B1CH          ;Länge des neuen Strings holen
  2A35 D5               PUSH    DE             ;und retten
  2A36 CF               RST     08H            ;Komma angegeben ?
  2A37 2C               DEFB    ','
  2A38 CD3723           CALL    2337H          ;Argument nach X holen
  2A3B CF               RST     08H            ;Klammer zu ?
  2A3C 29               DEFB    ')'
  2A3D E3               EX      (SP).HL        ;PTZ retten. Länge zurück
  2A3E E5               PUSH    HL             ;Länge retten
```

```
   2A3F E7                 RST     20H          ;TSTTYP, Stringargument ?
   2A40 2805               JR      Z,2A47H      ;Ja: weiter bei 2A47H
   2A42 CD1F2B             CALL    2B1FH        ;Nein: Ist die Zahl im Bereich
                                                ;von 0 bis 255 ? Nein: FC-Error
   2A45 1803               JR      2A4AH        ;Weiter bei 2A4AH (A = Zahl)

 ; String als 2. Argument angegeben

   2A47 CD132A             CALL    2A13H        ;ASCII-Wert des ersten Zeichens
                                                ;des Stringarguments nach A holen
   2A4A D1                 POP     DE           ;Länge zurück
   2A4B F5                 PUSH    AF           ;Dummy-PUSH wegen Sprung nach
                                                ;2A2BH nach Beendigung
   2A4C F5                 PUSH    AF           ;Zeichen retten
   2A4D 7B                 LD      A,E          ;A = Länge
   2A4E CD5728             CALL    2857H        ;Platz schaffen
   2A51 5F                 LD      E,A          ;E = Zähler
   2A52 F1                 POP     AF           ;Zeichen zurück
   2A53 1C                 INC     E            ;Zähler = 0 ?
   2A54 1D                 DEC     E
   2A55 28D4               JR      Z,2A2BH      ;Ja: String übernehmen
   2A57 2AD440             LD      HL,(40D4H)   ;Nein: HL = Zeiger auf freien
                                                ;Speicherplatz
   2A5A 77                 LD      (HL),A       ;Zeichen abspeichern
   2A5B 23                 INC     HL           ;Zeiger +1
   2A5C 1D                 DEC     E            ;Zähler -1, Zähler = 0 ?
   2A5D 20FB               JR      NZ,2A5AH     ;Nein: Nächstes Zeichen
   2A5F 18CA               JR      2A2BH        ;Ja: String übernehmen

 ; LEFT$
 ; I: (SP - 2) : Vektor des Stringarguments
 ;    (SP)     = Zahl

   2A61 CDDF2A             CALL    2ADFH        ;Klammer zu ?
                                                ;B = Zahl = Länge des neuen
                                                ;Strings
   2A64 AF                 XOR     A            ;Startpunkt auf 0 setzen

 ; Einsprung für RIGHT$
 ; I: A = Startpunkt des neuen Strings im Stringargument - 1
 ;    B = Länge des neuen Strings

   2A65 E3                 EX      (SP),HL      ;PTZ retten, HL : Vektor
   2A66 4F                 LD      C,A          ;C = A = Startpunkt des neuen
                                                ;Strings - 1
   2A67 3EE5               LD      A,0E5H       ;--

 ; Einsprung für MID$
 ; I: B = Länge des neuen Strings
 ;    C = Startpunkt des neuen Strings im Stringargument - 1

  *2A68  E5                PUSH    HL           ;Vektoradresse retten
```

```
: Teilstring aus dem Stringargument (in X) holen und wieder in X ablegen
: I: X = Stringargument
:    B = Länge des gewünschten Teilstrings
:    C = Offset zum Startpunkt des Teilstrings im Stringargument
: O: X = gewünschter Teilstring

 2A69 E5           PUSH   HL           ;Vektoradresse retten
 2A6A 7E           LD     A,(HL)       ;A = Länge des Stringarguments
 2A6B B8           CP     B            ;Soll der Teilstring länger als
                                       ;das Stringargument werden ?
 2A6C 3802         JR     C,2A70H      ;Ja: C auf 0 setzen, gesamtes
                                       ;Stringargument übernehmen
 2A6E 78           LD     A,B          ;Nein: A = gewünschte Länge
 2A6F 110E00       LD     DE,000EH     ;--
*2A70   0E00       LD     C,00H        ;Startpunkt = 0
 2A72 C5           PUSH   BC -         ;BC retten
 2A73 CDBF28       CALL   28BFH        ;A Bytes Platz schaffen
                                       ;DE = Zeiger auf freien Platz
 2A76 C1           POP    BC           ;BC zurück
 2A77 E1           POP    HL           ;Vektoradresse zurück
 2A78 E5           PUSH   HL           ;Vektoradresse retten
 2A79 23           INC    HL           ;Vektoradresse +1
 2A7A 46           LD     B,(HL)       ;HL = Zeiger auf Stringargument
 2A7B 23           INC    HL
 2A7C 66           LD     H,(HL)
 2A7D 68           LD     L,B
 2A7E 0600         LD     B,00H        ;BC = Offset zum Startpunkt des
                                       ;neuen Strings
 2A80 09           ADD    HL,BC        ;HL = Startadresse + Offset =
                                       ;Startadresse des neuen Strings
 2A81 44           LD     B,H          ;BC = Startadresse des neuen
                                       ;Strings
 2A82 4D           LD     C,L
 2A83 CD5A28       CALL   285AH        ;Länge und Adresse des neuen
                                       ;Strings in Stringtabelle ablegen
 2A86 6F           LD     L,A          ;L = Länge des neuen Strings
 2A87 CDCE29       CALL   29CEH        ;Entsprechende Zeichen des
                                       ;Stringarguments nach (DE)
                                       ;kopieren
 2A8A D1           POP    DE           ;Vektoradresse des String-
                                       ;arguments zurück nach DE
 2A8B CDDE29       CALL   29DEH        ;Stringargument aus Stringtabelle
                                       ;und Stringspeicher löschen ?
 2A8E C38428       JP     2884H        ;Neuen String übernehmen

: RIGHT$
: I: (SP - 2) : Vektor des Stringarguments
:    (SP)     = Zahl

 2A91 CDDF2A       CALL   2ADFH        ;Klammer zu ?, B = Zahl
 2A94 D1           POP    DE           ;Vektoradresse nach DE
 2A95 D5           PUSH   DE           ;zurück ins Stack
 2A96 1A           LD     A,(DE)       ;A = Länge des Stringarguments
 2A97 90           SUB    B            ;A = Länge - Zahl = Startpunkt
                                       ;des neuen Strings
 2A98 18CB         JR     2A65H        ;weiter bei 2A65H
```

```
; MID$ (rechts vom Gleichheitszeichen)
; (SP - 2) : Vektor des Stringarguments
; (SP)     = Zahl (Startpunkt des neuen Strings)

  2A9A EB          EX     DE,HL       ;HL = PTZ
  2A9B 7E          LD     A,(HL)      ;A = nächstes Zeichen
  2A9C CDE22A      CALL   2AE2H       ;Start nach B holen
  2A9F 04          INC    B           ;Startpunkt =  0 ?
  2AA0 05          DEC    B
  2AA1 CA4A1E      JP     Z,1E4AH     ;Ja: FC-Error
  2AA4 C5          PUSH   BC          ;Start retten
  2AA5 1EFF        LD     E,0FFH      ;E = Default Länge (Bei fehlender
                                      ;Längenangabe wird der gesamte
                                      ;String ab Start übernommen)
  2AA7 FE29        CP     29H         ;Ist das nächste Zeichen
                                      ;Klammer zu ?
  2AA9 2805        JR     Z,2AB0H     ;Ja: E ist Länge
  2AAB CF          RST    08H         ;Nein: Die Längenangabe muß durch
  2AAC 2C          DEFB   ','         ;ein Komma abgetrennt sein
  2AAD CD1C2B      CALL   2B1CH       ;Länge nach E holen
  2AB0 CF          RST    08H         ;Klammer geschlossen ?
  2AB1 29          DEFB   ')'
  2AB2 F1          POP    AF          ;A = Startpunkt
  2AB3 E3          EX     (SP),HL     ;PTZ retten. Vektoradresse zurück
  2AB4 01692A      LD     BC,2A69H    ;RET-Adr auf 2A69H setzen
  2AB7 C5          PUSH   BC
  2AB8 3D          DEC    A           ;A = Startpunkt - 1
  2AB9 BE          CP     (HL)        ;mit Länge des Stringarguments
                                      ;vergleichen
  2ABA 0600        LD     B,00H       ;Länge des zu erzeugenden
                                      ;Strings = 0
  2ABC D0          RET    NC          ;String mit der Länge 0 erzeugen
                                      ;wenn der angegebene Startpunkt
                                      ;größer als die Länge des String-
                                      ;arguments ist
  2ABD 4F          LD     C,A         ;C = Startpunkt - 1
  2ABE 7E          LD     A,(HL)      ;A = Länge der Stringarguments
  2ABF 91          SUB    C           ;- Startpunkt ergibt Restlänge
                                      ;des Stringarguments ab dem
                                      ;Startpunkt
  2AC0 BB          CP     E           ;Ist die gewünschte Länge größer
                                      ;als die Restlänge ?
  2AC1 47          LD     B,A         ;B = Restlänge
  2AC2 D8          RET    C           ;Ja: B ist Länge
  2AC3 43          LD     B,E         ;Nein: B mit gewünschter Länge
                                      ;laden
  2AC4 C9          RET                ;und damit weiterarbeiten
```

```
; VAL

  2AC5 CD072A          CALL    2A07H        ;Vektoradresse und Länge des
                                            ;Arguments holen. D = 00H
  2AC8 CAF827          JP      Z,27F8H      ;Ergebnis = 0 wenn die Länge des
                                            ;Stringarguments Null ist
  2ACB 5F              LD      E,A          ;E = Länge
  2ACC 23              INC     HL           ;Vektoradresse +1
  2ACD 7E              LD      A,(HL)       ;HL = Stringadresse
  2ACE 23              INC     HL
  2ACF 66              LD      H,(HL)
  2AD0 6F              LD      L,A
  2AD1 E5              PUSH    HL           ;Adresse retten
  2AD2 19              ADD     HL,DE        ;Endadresse des Strings errechnen
  2AD3 46              LD      B,(HL)       ;Nachfolgendes Zeichen holen
  2AD4 72              LD      (HL),D       ;und durch 00H ersetzen
  2AD5 E3              EX      (SP),HL      ;Endadresse retten, Startadresse
                                            ;zurück
  2AD6 C5              PUSH    BC           ;Zeichen retten
  2AD7 7E              LD      A,(HL)       ;A = 1. Zeichen
  2AD8 CD650E          CALL    0E65H        ;String decodieren, X ist Zahl
  2ADB C1              POP     BC           ;Zeichen zurück
  2ADC E1              POP     HL           ;Endadresse zurück
  2ADD 70              LD      (HL),B       ;Zeichen wieder einsetzen
  2ADE C9              RET

; UPRO für LEFT$, RIGHT$ und MID$
; Prüft ob die Klammer geschlossen wurde und holt die erste Zahl aus dem Stack

  2ADF EB              EX      DE,HL        ;HL = PTZ
  2AE0 CF              RST     08H          ;Klammer geschlossen
  2AE1 29              DEFB    ')'
  2AE2 C1              POP     BC           ;BC = RET-Adr
  2AE3 D1              POP     DE           ;DE = Zahl
  2AE4 C5              PUSH    BC           ;RET-Adr wieder ins Stack
  2AE5 43              LD      B,E          ;B = Zahl
  2AE6 C9              RET

; Gefundenes Token ist nicht im Bereich 80H bis BBH (siehe 1D67H ff)
; (kein eigenständiger Befehl)

  2AE7 FE7A            CP      7AH          ;Ist es MID$ ?
                                            ;(links vom Gleichheitszeichen !)
  2AE9 C29719          JP      NZ,1997H     ;Nein: SN-Error
  2AEC C3D941          JP      41D9H        ;Ja: DOS

; INP

  2AEF CD1F2B          CALL    2B1FH        ;Portadresse nach A holen
  2AF2 329440          LD      (4094H),A    ;und abspeichern
  2AF5 CD9340          CALL    4093H        ;INP ausführen
  2AF8 C3F827          JP      27F8H        ;Wert als INT übergeben
```

```
; OUT

  2AFB CD0E2B             CALL    2B0EH        ;Portadresse und Wert holen
  2AFE C39640             JP      4096H        ;OUT ausführen

; Argument bei (HL) decodieren und als INT-Wert nach X übergeben
; I: HL = PTZ
; O: DE = Zahl
;    A  = MSB der Zahl
;    Z  = 1 wenn Zahl < 256 (MSB = 0)

  2B01 D7                 RST     10H          ;PTZ erhöhen
  2B02 CD3723             CALL    2337H        ;Argument decodieren
  2B05 E5                 PUSH    HL           ;PTZ retten
  2B06 CD7F0A             CALL    0A7FH        ;HL = X = CINT(X)
  2B09 EB                 EX      DE,HL        ;DE = Zahl
  2B0A E1                 POP     HL           ;PTZ zurück
  2B0B 7A                 LD      A,D          ;A = MSB
  2B0C B7                 OR      A            ;MSB = 0 ?
  2B0D C9                 RET

; UPRO für OUT: Holt Portadresse, Wert und bereitet das RAM ab 4096H vor

  2B0E CD1C2B             CALL    2B1CH        ;Portadresse holen
  2B11 329440             LD      (4094H),A    ;Für INP
  2B14 329740             LD      (4097H),A    ;und OUT abspeichern
  2B17 CF                 RST     08H          ;Komma angegeben ?
  2B18 2C                 DEFB    ','
  2B19 1801               JR      2B1CH        ;Wert holen und Rücksprung

; Argument bei (HL) decodieren als INT-Wert nach X übergeben
; FC-Error wenn das Ergebnis nicht im Bereich von 0 bis 255 liegt
; I: HL = PTZ
; O: A  = Zahl
;    DE = Zahl

  2B1B D7                 RST     10H          ;PTZ erhöhen
  2B1C CD3723             CALL    2337H        ;Argument decodieren
  2B1F CD052B             CALL    2B05H        ;Im Bereich von 0 bis 255 ?
  2B22 C24A1E             JP      NZ,1E4AH     ;Nein: FC-Error
  2B25 2B                 DEC     HL           ;PTZ -1
  2B26 D7                 RST     10H          ;und wieder erhöhen
  2B27 7B                 LD      A,E          ;A = Zahl
  2B28 C9                 RET
```

```
: LLIST

  2B29 3E01          LD      A,01H         ;Ausgabeflag auf Druckerausgabe
  2B2B 329C40        LD      (409CH),A     ;setzen

: LIST

  2B2E C1            POP     BC            ;RET-Adr löschen
  2B2F CD101B        CALL    1B10H         ;Zeilennummern holen
  2B32 C5            PUSH    BC            ;ZP auf Startzeile retten

  2B33 21FFFF        LD      HL,0FFFFH     ;Aktuelle ZN auf 65535 setzen
  2B36 22A240        LD      (40A2H),HL
  2B39 E1            POP     HL            ;HL = ZP auf Zeile
  2B3A D1            POP     DE            ;DE = ZN der Endzeile
  2B3B 4E            LD      C,(HL)        ;BC = ZP auf die nächste Zeile
  2B3C 23            INC     HL
  2B3D 46            LD      B,(HL)
  2B3E 23            INC     HL
  2B3F 78            LD      A,B           ;Programmende erreicht ?
  2B40 B1            OR      C             ;ZP = 0000H ?
  2B41 CA191A        JP      Z,1A19H       ;Ja: Fertig
  2B44 CDDF41        CALL    41DFH         ;DOS
  2B47 CD9B1D        CALL    1D9BH         ;Shift-§ oder Break gedrückt ?
  2B4A C5            PUSH    BC            ;ZP retten
  2B4B 4E            LD      C,(HL)        ;BC = Zeilennummer der aktuellen
  2B4C 23            INC     HL            ;Zeile
  2B4D 46            LD      B,(HL)
  2B4E 23            INC     HL
  2B4F C5            PUSH    BC            ;Zeilennummer retten
  2B50 E3            EX      (SP),HL       ;Zeiger retten, ZN zurück
  2B51 EB            EX      DE,HL         ;DE = Zeilennummer der aktuellen
                                           ;Zeile, HL = Zeilennummer der
                                           ;Endzeile
  2B52 DF            RST     18H           ;Ist die aktuelle Zeilennummer
                                           ;schon größer als die der End-
                                           ;zeile ?
  2B53 C1            POP     BC            ;Zeiger auf die aktuelle Zeile
                                           ;zurück nach BC
  2B54 DA181A        JP      C,1A18H       ;Ja: Fertig
  2B57 E3            EX      (SP),HL       ;Nein: Endzeilennummer retten
                                           ;HL = ZP auf die nächste Zeile
  2B58 E5            PUSH    HL            ;ZP auf die nächste Zeile retten
  2B59 C5            PUSH    BC            ;ZP auf die aktuelle Zeile retten
  2B5A EB            EX      DE,HL         ;HL = aktuelle Zeilennummer
  2B5B 22EC40        LD      (40ECH),HL    ;als '.'-Zeile abspeichern
  2B5E CDAF0F        CALL    0FAFH         ;Zeilennummer ausgeben
  2B61 3E20          LD      A,20H         ;A = Leerzeichen
  2B63 E1            POP     HL            ;Zeiger auf die aktuelle Zeile
                                           ;zurück nach HL
```

```
 2B64 CD2A03            CALL    032AH          :Leerzeichen ausgeben
 2B67 CD7E2B            CALL    2B7EH          :Zeile ab (HL) decodieren und im
                                               :Zeilenbuffer ablegen
 2B6A 2AA740            LD      HL,(40A7H)     :HL = Startadresse des Zeilen-
                                               :buffers
 2B6D CD752B            CALL    2B75H          :Zeile ausgeben
 2B70 CDFE20            CALL    20FEH          :Neue Zeile auf dem Bildschirm
                                               :beginnen
 2B73 18BE              JR      2B33H          :Nächste Zeile bearbeiten

; UPRO für LIST
; Text ab (HL) ausgeben. 00H ist Textende
; I: HL = Zeiger auf auszugebenden Text
; O: HL = Zeiger auf Textende (00H)

 2B75 7E                LD      A,(HL)         :A = Nächstes Zeichen
 2B76 B7                OR      A              :Zeilenende erreicht ?
 2B77 C8                RET     Z              :Ja: Fertig
 2B78 CD2A03            CALL    032AH          :Nein: Zeichen ausgeben
 2B7B 23                INC     HL             :Zeiger +1
 2B7C 18F7              JR      2B75H          :Nächstes Zeichen

; UPRO für LIST und EDIT
; Zeile ab (HL) decodieren und im Zeilenbuffer ablegen
; I: HL = Zeiger auf Programmtext

 2B7E E5                PUSH    HL             :Zeiger retten
 2B7F 2AA740            LD      HL,(40A7H)     :HL = Startadresse des Zeilen-
                                               :buffers
 2B82 44                LD      B,H            :BC = Startadresse des Zeilen-
 2B83 4D                LD      C,L            :buffers
 2B84 E1                POP     HL             :Zeiger zurück
 2B85 16FF              LD      D,0FFH         :D = Zähler (maximale Länge =
                                               :255 Zeichen)
 2B87 1803              JR      2B8CH          :weiter bei 2B8CH

; Nächstes Zeichen decodieren

 2B89 03                INC     BC             :Zeiger +1
 2B8A 15                DEC     D              :Zähler -1, maximale Länge
                                               :erreicht ?
 2B8B C8                RET     Z              :Ja: Fertig
```

```
; Zeile ab (HL) decodieren und in (BC) ablegen
; I: HL = Zeiger auf Programmtext
;    BC = Zeiger auf Buffer
;    D  = maximale Zeilenlänge

  2B8C 7E           LD     A,(HL)      :A = nächstes Zeichen
  2B8D B7           OR     A           :Zeilenende erreicht ?
  2B8E 23           INC    HL          :Zeiger +1
  2B8F 02           LD     (BC),A      :Zeichen abspeichern
  2B90 C8           RET    Z           :Ja: Fertig
  2B91 F2D23F       JP     P,3FD2H     :weiter bei 3FD2H wenn kein Token
                                       :gefunden wurde
  2B94 FEFB         CP     0FBH        :"'" gefunden (REM) ?
  2B96 2008         JR     NZ,2BA0H    :Nein: weiter bei 2BA0H
  2B98 0B           DEC    BC          :Ja: die letzten 4 Zeichen wieder
  2B99 0B           DEC    BC          :löschen da das Apostroph als
  2B9A 0B           DEC    BC          : ::REM' abgespeichert wird
  2B9B 0B           DEC    BC
  2B9C 14           INC    D           :Zähler +4
  2B9D 14           INC    D
  2B9E 14           INC    D
  2B9F 14           INC    D
  2BA0 FE95         CP     95H         :'ELSE'-Token gefunden ?
  2BA2 CC240B       CALL   Z,0B24H     :Ja: Bufferzeiger -1
                                       :('ELSE' wird als ':ELSE'
                                       :abgespeichert)
  2BA5 D67F         SUB    7FH         :A = Tokenwert - 7FH
  2BA7 E5           PUSH   HL          :Zeiger retten
  2BA8 5F           LD     E,A         :E = Tokenwert
  2BA9 CDAD39       CALL   39ADH       :Colour-Token gefunden ?
                                       :HL zeigt auf den Anfang der
                                       :entsprechenden Keywordtabelle
  2BAC 7E           LD     A,(HL)      :A = nächstes Zeichen
  2BAD B7           OR     A           :nächstes Keyword erreicht ?
  2BAE 23           INC    HL          :Zeiger +1
  2BAF F2AC2B       JP     P,2BACH     :Nein: HL bis zum nächsten Key-
                                       :word erhöhen
  2BB2 1D           DEC    E           :Ja: Tokenzähler -1
  2BB3 20F7         JR     NZ,2BACH    :Zeiger weiter erhöhen bis
                                       :entsprechendes Keyword
                                       :erreicht ist
  2BB5 E67F         AND    7FH         :Bit7 ausblenden
  2BB7 02           LD     (BC),A      :Zeichen ablegen
  2BB8 03           INC    BC          :Bufferzeiger +1
  2BB9 15           DEC    D           :Zähler -1
  2BBA CAD828       JP     Z,28D8H     :Fertig wenn die maximale Zeilen-
                                       :länge erreicht wurde
  2BBD 7E           LD     A,(HL)      :A = nächstes Zeichen
  2BBE 23           INC    HL          :Tabellenzeiger +1
  2BBF B7           OR     A           :nächstes Keyword erreicht
  2BC0 F2B72B       JP     P,2BB7H     :Nein: Zeichen im Buffer ablegen
  2BC3 E1           POP    HL          :Ja: Zeilenzeiger zurück
  2BC4 18C6         JR     2B8CH       :Nächstes Zeichen bearbeiten
```

```
; DELETE

  2BC6 CD101B        CALL   1B10H          ;Start- und Endzeile holen
  2BC9 D1            POP    DE             ;DE = ZN der Endzeile
  2BCA C5            PUSH   BC             ;Zeiger auf die Startzeile
  2BCB C5            PUSH   BC             ;zweimal retten
  2BCC CD2C1B        CALL   1B2CH          ;Endzeile suchen
  2BCF 3005          JR     NC,2BD6H       ;FC-Error wenn die Endzeile nicht
                                           ;existiert
  2BD1 54            LD     D,H            ;DE = Zeiger der Endzeile (zeigt
  2BD2 5D            LD     E,L            ;auf die der Endzeile folgende
                                           ;Zeile)
  2BD3 E3            EX     (SP),HL        ;Zeiger der Endzeile retten
                                           ;HL = Zeiger auf die Startzeile
  2BD4 E5            PUSH   HL             ;Zeiger auf die Startzeile retten
  2BD5 DF            RST    18H            ;Beide Zeiger vergleichen
  2BD6 D24A1E        JP     NC,1E4AH       ;FC-Error wenn die Zeilen nicht
                                           ;aufeinanderfolgend angegeben
                                           ;wurden
  2BD9 212919        LD     HL,1929H       ;HL = Zeiger auf Text 'READY'
  2BDC CDA728        CALL   28A7H          ;Text ausgeben
  2BDF C1            POP    BC             ;Zeiger auf die Startzeile zurück
  2BE0 21E81A        LD     HL,1AE8H       ;RET-Adr auf 1AE8H setzen (POP DE
                                           ;und Zeiger erneuern)
  2BE3 E3            EX     (SP),HL        ;Zeiger der Endzeile zurück

; Zeile(n) löschen
; I: BC = Zeiger auf die Startzeile
;    HL = Zeiger der Endzeile (zeigt auf nächste Zeile nach der Endzeile)

  2BE4 EB            EX     DE,HL          ;DE = Zeiger auf die Endzeile
  2BE5 2AF940        LD     HL,(40F9H)     ;HL = Endadresse des Programms
  2BE8 1A            LD     A,(DE)         ;Nächste Zeile
  2BE9 02            LD     (BC),A         ;über die zu löschende Zeile
                                           ;kopieren
  2BEA 03            INC    BC             ;Zeiger +1
  2BEB 13            INC    DE             ;Zeiger +1
  2BEC DF            RST    18H            ;Programmende erreicht ?
  2BED 20F9          JR     NZ,2BE8H       ;Nein: weiter kopieren
  2BEF 60            LD     H,B            ;HL = neue Endadresse des
  2BF0 69            LD     L,C            ;Programms
  2BF1 22F940        LD     (40F9H),HL     ;abspeichern
  2BF4 C9            RET
```

: CSAVE

```
2BF5 CD3723        CALL   2337H          ;Filenamen holen
2BF8 E5            PUSH   HL             ;PTZ retten
2BF9 CD132A        CALL   2A13H          ;1. Zeichen des Stringarguments
                                         ;nach A holen
2BFC F5            PUSH   AF             ;Zeichen retten
2BFD C5            PUSH   BC             ;Register retten
2BFE D5            PUSH   DE
2BFF E5            PUSH   HL
2C00 CD3F02        CALL   023FH          ;Leader & Sync schreiben
2C03 E1            POP    HL             ;Register zurück
2C04 D1            POP    DE
2C05 C1            POP    BC
2C06 F1            POP    AF             ;Zeichen zurück
2C07 1A            LD     A,(DE)         ;1. Zeichen nochmal laden
2C08 CD1F02        CALL   021FH          ;Filename schreiben
2C0B 2AA440        LD     HL,(40A4H)     ;HL = Startadresse des Programms
2C0E EB            EX     DE,HL          ;DE = HL
2C0F 2AF940        LD     HL,(40F9H)     ;HL = Endadresse des Programms
2C12 1A            LD     A,(DE)         ;Zeichen holen
2C13 13            INC    DE             ;Zeiger +1
2C14 CD1F02        CALL   021FH          ;Zeichen auf Cassette schreiben
2C17 DF            RST    18H            ;Programmende erreicht ?
2C18 20F8          JR     NZ,2C12H       ;Nein: nächstes Zeichen
2C1A 00            NOP                   ;--
2C1B 00            NOP
2C1C 00            NOP
2C1D E1            POP    HL             ;PTZ zurück
2C1E C9            RET
```

: CLOAD

```
2C1F 00            NOP                   ;--
2C20 00            NOP
2C21 00            NOP
2C22 00            NOP
2C23 00            NOP
2C24 00            NOP
2C25 00            NOP
2C26 00            NOP
2C27 AF            XOR    A              ;A = 00H
2C28 012F23        LD     BC,232FH       ;--
```

; Einsprung für VERIFY (siehe 3F34H)

```
*2C29   2F        CPL                   ;A <> 00H
*2C2A     23      INC     HL            ;PTZ +1
 2C2B F5          PUSH    AF            ;Flag retten
 2C2C 2B          DEC     HL            ;PTZ -1
 2C2D D7          RST     10H           ;Filenamen angegeben ?
 2C2E 3E00        LD      A,00H         ;A = Default-Filename
 2C30 2807        JR      Z,2C39H       ;Nein: weiter bei 2C39H
 2C32 CD3723      CALL    2337H         ;Ja: Filenamen
 2C35 CD132A      CALL    2A13H         ;nach A holen
 2C38 1A          LD      A,(DE)
 2C39 6F          LD      L,A           ;L = Filename
 2C3A F1          POP     AF            ;Flag zurück
 2C3B B7          OR      A             ;CLOAD ?
 2C3C 67          LD      H,A           ;H = Flag
 2C3D 222141      LD      (4121H),HL    ;Flag und Filenamen abspeichern
 2C40 CC4D1B      CALL    Z,1B4DH       ;Ja: NEW
 2C43 2A2141      LD      HL,(4121H)    ;Flag und Filenamen zurück
 2C46 EB          EX      DE,HL         ;D = Flag, E = Filename
 2C47 F5          PUSH    AF            ;Register retten
 2C48 C5          PUSH    BC
 2C49 D5          PUSH    DE
 2C4A E5          PUSH    HL
 2C4B CD4C02      CALL    024CH         ;Leader und Sync suchen
 2C4E E1          POP     HL            ;Register zurück
 2C4F D1          POP     DE
 2C50 C1          POP     BC
 2C51 F1          POP     AF
 2C52 CDED01      CALL    01EDH         ;1 Byte lesen (Filename)
 2C55 1C          INC     E             ;Filename angegeben ?
 2C56 1D          DEC     E
 2C57 2803        JR      Z,2C5CH       ;Nein: weiter bei 2C5CH
 2C59 BB          CP      E             ;Ja: Richtiges File gefunden ?
 2C5A 2037        JR      NZ,2C93H      ;Nein: Beim nächsten File
                                        ;probieren
 2C5C 2AA440      LD      HL,(40A4H)    ;HL = Startadresse des Programms
 2C5F 0603        LD      B,03H         ;B = Zähler (3 mal 00H kenn-
                                        ;zeichnet das Programmende)
 2C61 CDED01      CALL    01EDH         ;1 Byte lesen
 2C64 5F          LD      E,A           ;E = Byte
 2C65 96          SUB     (HL)          ;steht dieses Byte im Speicher ?
                                        ;Ja: Dann ist A = 00H
 2C66 A2          AND     D             ;Mit Flag verknüpfen
                                        ;Ja: A ist 00H bei CLOAD oder
                                        ;wenn bei VERIFY das Byte schon
                                        ;im Speicher stand.
                                        ;War jedoch bei VERIFY das
                                        ;Ergebnis des Vergleichs <> 0 so
                                        ;ist A <> 0
```

```
2C67 2021           JR      NZ,2C8AH        ;weiter bei 2C8AH wenn A <> 0 ist
2C69 73             LD      (HL),E          ;Byte abspeichern
2C6A CD6C19         CALL    196CH           ;Noch Speicher frei ?
2C6D 7E             LD      A,(HL)          ;War das gelesene Byte = 00H ?
2C6E B7             OR      A
2C6F 23             INC     HL              ;Zeiger +1
2C70 20ED           JR      NZ,2C5FH        ;Nein: nächstes Byte
2C72 CDE401         CALL    01E4H           ;Ja: * blinken
2C75 10EA           DJNZ    2C61H           ;Zähler -1, nächstes Byte lesen

2C77 22F940         LD      (40F9H),HL      ;Neues Programmende abspeichern
2C7A 212919         LD      HL,1929H        ;Text 'READY'
2C7D CDA728         CALL    28A7H           ;ausgeben
2C80 00             NOP                     ;--
2C81 00             NOP
2C82 00             NOP
2C83 2AA440         LD      HL,(40A4H)      ;HL = Startadresse des Programms
2C86 E5             PUSH    HL              ;HL retten
2C87 C3E81A         JP      1AE8H           ;Alle Zeiger im Programmtext
                                            ;erneuern

; Fehler bei VERIFY

2C8A 21A52C         LD      HL,2CA5H        ;HL : Text 'BAD'
2C8D CD7935         CALL    3579H           ;Text und Ton ausgeben
2C90 C3181A         JP      1A18H           ;Zurück zum aktiven Befehlsmodus

; Filename nicht gefunden

2C93 322644         LD      (4426H),A       ;Gefundenen Filename anzeigen
2C96 0603           LD      B,03H           ;Ende des Programms suchen
                                            ;(3 x 00H)
2C98 CDED01         CALL    01EDH           ;Byte lesen
2C9B B7             OR      A               ;00H gefunden ?
2C9C 20F8           JR      NZ,2C96H        ;Nein: nächstes Byte
2C9E 10F8           DJNZ    2C98H           ;Ja: Zähler -1, nächstes Byte
                                            ;lesen
2CA0 00             NOP                     ;--
2CA1 00             NOP
2CA2 00             NOP
2CA3 18A2           JR      2C47H           ;CLOAD neu versuchen

; Text 'BAD'

2CA5 42
2CA6 41
2CA7 44
2CA8 0D
2CA9 00
```

```
: X = PEEK ( X )

  2CAA CD7F0A            CALL    0A7FH          :HL = X = CINT(X) = Adresse
  2CAD 7E                LD      A,(HL)         :A = Inhalt des Speicherplatzes
  2CAE C3F827            JP      27F8H          :A als INT übergeben

; POKE

  2CB1 CD022B            CALL    2B02H          :Adresse nach DE holen
  2CB4 D5                PUSH    DE             :Adresse retten
  2CB5 CF                RST     08H            :Komma angegeben ?
  2CB6 2C                DEFB    ','
  2CB7 CD1C2B            CALL    2B1CH          :Wert holen
  2CBA D1                POP     DE             :Adresse zurück
  2CBB 12                LD      (DE),A         :Wert im Speicher ablegen
  2CBC C9                RET     -

; PRINT USING

  2CBD CD3823            CALL    2338H          :Formatstring holen
  2CC0 CDF40A            CALL    0AF4H          :TM-Error wenn kein String
                                                :gefunden wurde
  2CC3 CF                RST     08H            :Danach muß ':' folgen
  2CC4 3B                DEFB    ';'
  2CC5 EB                EX      DE,HL          :DE = PTZ
  2CC6 2A2141            LD      HL,(4121H)     :HL : Vektor des Strings
  2CC9 1808              JR      2CD3H          :weiter bei 2CD3H

; Wiedereinsprung wenn mehrere Zahlenwerte mit dem selben Formatstring
; ausgegeben werden sollen

  2CCB 3ADE40            LD      A,(40DEH)      :A = Zeichen nach dem Trennungs-
                                                :zeichen zur nächsten Variablen
  2CCE B7                OR      A              :Steht nach dem Trennungszeichen
                                                :noch eine Variable ?
  2CCF 280C              JR      Z,2CDDH        :Nein: FC-Error
  2CD1 D1                POP     DE             :Vektoradresse des Formatstrings
                                                :zurück
  2CD2 EB                EX      DE,HL          :HL : Vektor, DE = PTZ

; PRINT USING ausführen
; I: DE = PTZ auf auszugebende Variable
;    HL : Vektor des Formatstrings

  2CD3 E5                PUSH    HL             :Vektoradresse retten
  2CD4 AF                XOR     A              :A = 0
  2CD5 32DE40            LD      (40DEH),A      :Zeichen löschen
  2CD8 BA                CP      D              :CY = 1, Z = 0
  2CD9 F5                PUSH    AF             :AF retten
  2CDA D5                PUSH    DE             :PTZ retten
  2CDB 46                LD      B,(HL)         :B = Länge des Formatstrings
  2CDC B0                OR      B              :Nullstring ?
  2CDD CA4A1E            JP      Z,1E4AH        :Ja: FC-Error
```

```
2CE0 23          INC    HL          ;Vektoradresse +1
2CE1 4E          LD     C,(HL)      ;HL = Stringzeiger
2CE2 23          INC    HL
2CE3 66          LD     H,(HL)
2CE4 69          LD     L,C
2CE5 181C        JR     2D03H       ;weiter bei 2D03H

; '%' gefunden
; I: B  = Restlänge des Strings
;    HL = Zeiger auf String (auf nächstes Zeichen hinter '%')
; O: C  = Anzahl der zwischen den beiden '%' gefundenen Leerzeichen + 2

2CE7 58          LD     E,B         ;Zähler nach E retten
2CE8 E5          PUSH   HL          ;Zeiger retten
2CE9 0E02        LD     C,02H       ;C = Zähler für Leerzeichen
                                    ;(2 für die beiden Begrenzungs-
                                    ;zeichen)
2CEB 7E          LD     A,(HL)      ;Nächstes Zeichen holen
2CEC 23          INC    HL          ;Zeiger +1
2CED FE25        CP     25H         ;2. '%'-Zeichen gefunden ?
2CEF CA172E      JP     Z,2E17H     ;Ja: Fertig
2CF2 FE20        CP     20H         ;Leerzeichen gefunden ?
2CF4 2003        JR     NZ,2CF9H    ;Nein: '%' als Textzeichen über-
                                    ;nehmen (Zwischen den beiden
                                    ;'%'-Zeichen dürfen nur Leer-
                                    ;zeichen stehen)
2CF6 0C          INC    C           ;Zähler +1
2CF7 10F2        DJNZ   2CEBH       ;Nächstes Zeichen holen
                                    ;Stringzähler -1
2CF9 E1          POP    HL          ;Stringzeiger zurück wenn kein
                                    ;2. '%'-Zeichen gefunden wurde
2CFA 43          LD     B,E         ;Alten Zählerstand zurück
2CFB 3E25        LD     A,25H       ;und '%'-Zeichen als Textzeichen
                                    ;übernehmen

; Das gefundene Zeichen ist kein Formatierungszeichen sondern Teil eines
; auszugebenden Textes

2CFD CD492E      CALL   2E49H       ;'+' ausgeben wenn D <> 0
2D00 CD2A03      CALL   032AH       ;Zeichen ausgeben

; Formatstring bearbeiten
; I: (SP - 4) : Vektor des Formatstrings
;    (SP - 2) = AF: A = 00H, CY = 1, Z = 0 (siehe 2CD9H)
;    (SP)     = PTZ
;     B       = Länge des Strings (bzw. Restlänge)
;     HL      = Zeiger auf String

2D03 AF          XOR    A           ;A = 00H
2D04 5F          LD     E,A         ;Anzahl der auszugebenden
                                    ;Vorkommastellen auf 0 setzen
2D05 57          LD     D,A         ;Formatbyte auf 0 setzen
                                    ;(Aufbau des Formatbytes
                                    ;siehe 0FBEH)
```

```
2D06 CD492E     CALL  2E49H      ;'+' ausgeben wenn D <> 0
2D09 57         LD    D.A        ;Formatbyte nach D
2D0A 7E         LD    A.(HL)     ;nächstes Zeichen holen
2D0B 23         INC   HL         ;Zeiger +1
2D0C FE21       CP    21H        ;'!' gefunden ?
2D0E CA142E     JP    Z.2E14H    ;Ja: weiter bei 2E14H
2D11 FE23       CP    23H        ;'#' gefunden ?
2D13 2837       JR    Z.2D4CH    ;Ja: weiter bei 2D4CH
2D15 05         DEC   B          ;Zähler -1. Stringende erreicht ?
2D16 CAFE2D     JP    Z.2DFEH    ;Ja: weiter bei 2DFEH
2D19 FE2B       CP    2BH        ;'+' gefunden ?
2D1B 3E08       LD    A.08H      ;Bit 3 des Formatbytes setzen
2D1D 28E7       JR    Z.2D06H    ;Ja: nächstes Zeichen holen
2D1F 2B         DEC   HL         ;Zeiger -1
2D20 7E         LD    A.(HL)     ;Zeichen nochmal holen
2D21 23         INC   HL         ;Zeiger +1
2D22 FE2E       CP    2EH        ;'.' gefunden ?
2D24 2840       JR    Z.2D66H    ;Ja: weiter bei 2D66H
2D26 FE25       CP    25H        ;'%' gefunden ?
2D28 28BD       JR    Z.2CE7H    ;Ja: weiter bei 2CE7H
2D2A BE         CP    (HL)       ;Zweimal das gleiche Zeichen ?
2D2B 20D0       JR    NZ.2CFDH   ;Nein: Zeichen nicht erkannt
2D2D FE24       CP    24H        ;'$$' gefunden ?
2D2F 2814       JR    Z.2D45H    ;Ja: weiter bei 2D45H
2D31 FE2A       CP    2AH        ;'**' gefunden ?
2D33 20C8       JR    NZ.2CFDH   ;Nein: Zeichen nicht erkannt

; '**' gefunden

2D35 78         LD    A.B        ;Restlänge nach A
2D36 FE02       CP    02H        ;weniger als 2 Zeichen übrig ?
2D38 23         INC   HL         ;Zeiger +1
2D39 3803       JR    C.2D3EH    ;Ja: weiter bei 2D3EH
2D3B 7E         LD    A.(HL)     ;nächstes Zeichen holen
2D3C FE24       CP    24H        ;Ist es '$' ?
2D3E 3E20       LD    A.20H      ;Bit 5 (zur '*'-Ausgabe) setzen
2D40 2007       JR    NZ.2D49H   ;Nein: Nur '**' gefunden
2D42 05         DEC   B          ;Ja: '**$' gefunden. Zähler -1
2D43 1C         INC   E          ;Vorkommastellen +1
2D44 FEAF       CP    0AFH       ;--

; '$$' gefunden

*2D45  AF       XOR   A          ;A = 0
2D46 C610       ADD   A.10H      ;Bit 4 (für '$' vor Zahl) setzen
2D48 23         INC   HL         ;Zeiger +1
2D49 1C         INC   E          ;Vorkommastellen +1
2D4A 82         ADD   A.D        ;A mit altem Formatbyte ver-
                                 ;knüpfen
2D4B 57         LD    D.A        ;D = Formatbyte
```

```
; '#' gefunden

2D4C 1C          INC   E          ;Vorkommastellen +1
2D4D 0E00        LD    C,00H      ;Zähler für die Textausgabe
                                  ;löschen
2D4F 05          DEC   B          ;Stringzähler -1
2D50 2847        JR    Z,2D99H    ;weiter bei 2D99H wenn es das
                                  ;letzte Zeichen war
2D52 7E          LD    A,(HL)     ;nächstes Zeichen holen
2D53 23          INC   HL         ;Zeiger +1
2D54 FE2E        CP    2EH        ;'.' (Dezimalpunkt) gefunden ?
2D56 2818        JR    Z,2D70H    ;Ja: weiter bei 2D70H
2D58 FE23        CP    23H        ;'#' gefunden ?
2D5A 28F0        JR    Z,2D4CH    ;Ja: wieder zurück nach 2D4CH
2D5C FE2C        CP    2CH        ;',' (Tausendertrennung)
                                  ;gefunden ?
2D5E 201A        JR    NZ,2D7AH   ;Nein: weiter bei 2D7AH

; ',' gefunden

2D60 7A          LD    A,D        ;Bit 6 des Formatbytes
2D61 F640        OR    40H        ;setzen
2D63 57          LD    D,A
2D64 18E6        JR    2D4CH      ;Nächstes Zeichen holen

; '.' vor '#' gefunden

2D66 7E          LD    A,(HL)     ;A = nächstes Zeichen
2D67 FE23        CP    23H        ;Ist das nächste Zeichen
                                  ;ein '#' ?
2D69 3E2E        LD    A,2EH      ;A = '.'
2D6B 2090        JR    NZ,2CFDH   ;Nein: '.' als Textzeichen
                                  ;übernehmen
2D6D 0E01        LD    C,01H      ;Zähler der Nachkommastellen auf
                                  ;1 setzen (für den Dezimalpunkt)
2D6F 23          INC   HL         ;Zeiger +1

; '.' nach '#' gefunden

2D70 0C          INC   C          ;Nachkommastellen +1
2D71 05          DEC   B          ;Zähler -1
2D72 2825        JR    Z,2D99H    ;weiter bei 2D99H wenn String
                                  ;zuende
2D74 7E          LD    A,(HL)     ;Nächstes Zeichen holen
2D75 23          INC   HL         ;Zeiger +1
2D76 FE23        CP    23H        ;Ist es ein '#'
2D78 28F6        JR    Z,2D70H    ;Ja: Nachkommastellen +1
```

```
: Soll die Zahl im Exponentialformat ausgegeben werden ?
: (Viermal Hochpfeil hintereinander)

  2D7A D5          PUSH  DE          :Formatbyte und Vorkommastellen-
                                     :zähler retten
  2D7B 11972D      LD    DE.2D97H    :RET-Adr auf 2D97H setzen
  2D7E D5          PUSH  DE
  2D7F 54          LD    D.H         :DE = Stringzeiger
  2D80 5D          LD    E.L
  2D81 FE5B        CP    5BH         :Wurde ein Hochpfeil gefunden ?
  2D83 C0          RET   NZ          :Nein: RET nach 2D97H
  2D84 BE          CP    (HL)        :2 x Hochpfeil ?
  2D85 C0          RET   NZ          :Nein: RET nach 2D97H
  2D86 23          INC   HL          :Zeiger +1
  2D87 BE          CP    (HL)        :3 x Hochpfeil ?
  2D88 C0          RET   NZ -        :Nein: RET nach 2D97H
  2D89 23          INC   HL          :Zeiger +1
  2D8A BE          CP    (HL)        :4 x Hochpfeil ?
  2D8B C0          RET   NZ          :Nein: RET nach 2D97H
  2D8C 23          INC   HL          :Zeiger +1
  2D8D 78          LD    A.B         :A = Stringzähler
  2D8E D604        SUB   04H         :4 abziehen. Überlauf ?
  2D90 D8          RET   C           :Ja: Die 4 Hochpfeile gehörten
                                     :nicht alle zum Formatstring

: Viermal Hochpfeil gefunden: Zahl im Exponentialformat ausgeben

  2D91 D1          POP   DE          :RET-Adr löschen (2D97H)
  2D92 D1          POP   DE          :Formatbyte und Anzahl der
                                     :Vorkommastellen zurück
  2D93 47          LD    B.A         :B = Stringzähler
  2D94 14          INC   D           :Bit 0 (für Exponentialausgabe)
                                     :setzen
  2D95 23          INC   HL          :Zeiger +1
  2D96 CAEBD1      JP    Z.0D1EBH    :Wird nie ausgeführt da Z <> 0
                                     :(wegen 2D94H)

: Keine Exponentialausgabe

 *2D97   EB        EX    DE.HL       :HL = Alter Stringzeiger
 *         D1      POP   DE          :Formatbyte und Anzahl der Vor-
                                     :kommastellen zurück

: Letztes Zeichen des Strings war '.' oder '#'

  2D99 7A          LD    A.D         :A = Formatbyte
  2D9A 2B          DEC   HL          :Zeiger -1
  2D9B 1C          INC   E           :Vorkommastellen +1
  2D9C E608        AND   08H         :Positives Vorzeichen ausgeben ?
                                     :(Bit 3 gesetzt ?)
  2D9E 2015        JR    NZ.2DB5H    :Ja: weiter bei 2DB5H
  2DA0 1D          DEC   E           :Nein: Vorkommastellen -1
```

```
2DA1 78          LD     A,B          ;Noch Stringzeichen übrig ?
2DA2 B7          OR     A
2DA3 2810        JR     Z,2DB5H      ;Nein: weiter bei 2DB5H
2DA5 7E          LD     A,(HL)       ;Ja: A = nächstes Zeichen
2DA6 D62D        SUB    2DH          ;'-' gefunden ?
2DA8 2806        JR     Z,2DB0H      ;Ja: weiter bei 2DB0H
2DAA FEFE        CP     0FEH         ;'+' gefunden ? (FEH + 2DH = 2BH)
2DAC 2007        JR     NZ,2DB5H     ;Nein: weiter bei 2DB5H
2DAE 3E08        LD     A,08H        ;Ja: Bit 3 setzen (Vorzeichen
                                     ;angeben)
2DB0 C604        ADD    A,04H        ;Bit 2 setzen (Vorzeichen auch
                                     ;hinter der Zahl ausgeben)
2DB2 82          ADD    A,D          ;mit Formatbyte verknüpfen
2DB3 57          LD     D,A          ;Formatbyte zurückschreiben
2DB4 05          DEC    B            ;Stringzähler -1

; Formatstring fertig bearbeitet
; I: B = Stringzähler
;    C = Anzahl der Nachkommastellen + 1 (für Dezimalpunkt)
;    D = Formatbyte
;    E = Anzahl der Vorkommastellen

2DB5 E1          POP    HL           ;PTZ zurück
2DB6 F1          POP    AF           ;Flag zurück (siehe 2CD9H)
2DB7 2850        JR     Z,2E09H      ;Fertig wenn Flag = 0
2DB9 C5          PUSH   BC           ;Register retten
2DBA D5          PUSH   DE
2DBB CD3723      CALL   2337H        ;Auszugebenden Zahlenwert holen
2DBE D1          POP    DE           ;Register zurück
2DBF C1          POP    BC
2DC0 C5          PUSH   BC           ;Stringzähler retten
2DC1 E5          PUSH   HL           ;PTZ retten
2DC2 43          LD     B,E          ;B = Anzahl der Vorkommastellen
2DC3 78          LD     A,B          ;A = Vorkommastellen
2DC4 81          ADD    A,C          ;+ Anzahl der Nachkommastellen
2DC5 FE19        CP     19H          ;mehr als 24 Stellen ausgeben ?
2DC7 D24A1E      JP     NC,1E4AH     ;Ja: FC-Error, die maximale
                                     ;Stellenzahl ist 24
                                     ;(1 Stelle  Vorzeichen
                                     ;17 Stellen DBL-Format
                                     ; 1 Stelle  Dezimalpunkt
                                     ; 4 Stellen Exponentialausgabe
                                     ; 1 Stelle  Vorzeichen nach Zahl)
2DCA 7A          LD     A,D          ;A = Formatbyte
2DCB F680        OR     80H          ;Bit 7 setzen (Formatierung
                                     ;durchführen)
2DCD CDBE0F      CALL   0FBEH        ;Zahl in X in formatierten String
                                     ;ab 4130H ablegen. HL ist Zeiger
                                     ;auf diesen String
2DD0 CDA728      CALL   28A7H        ;Zahl(enstring) ausgeben
2DD3 E1          POP    HL           ;PTZ zurück
2DD4 2B          DEC    HL           ;PTZ -1
2DD5 D7          RST    10H          ;nächstes Zeichen holen
2DD6 37          SCF                 ;CY = 1
2DD7 280D        JR     Z,2DE6H      ;Sprung wenn Anweisungsende
                                     ;erreicht wurde
```

```
2DD9 32DE40          LD      (40DEH),A    ;nächstes Zeichen abspeichern
                                          ;(siehe 2CCBH ff)
2DDC FE3B            CP      3BH          ;';' gefunden ?
2DDE 2805            JR      Z,2DE5H      ;Ja: Zeichen ok
2DE0 FE2C            CP      2CH          ;',' gefunden ?
2DE2 C29719          JP      NZ,1997H     ;Nein: SN-Error
2DE5 D7              RST     10H          ;PTZ erhöhen
2DE6 C1              POP     BC           ;Stringzähler zurück
2DE7 EB              EX      DE,HL        ;DE = PTZ
2DE8 E1              POP     HL           ;HL : Vektor des Formatstrings
2DE9 E5              PUSH    HL           ;Vektoradresse retten
2DEA F5              PUSH    AF           ;nächstes Zeichen retten
2DEB D5              PUSH    DE           ;PTZ retten
2DEC 7E              LD      A,(HL)       ;A = Länge des Formatstrings
2DED 90              SUB     B            ;Restlänge abziehen
2DEE 23              INC     HL           ;Vektoradresse +1
2DEF 4E              LD      C,(HL)       ;HL = Zeiger auf den Formatstring
2DF0 23              INC     HL
2DF1 66              LD      H,(HL)
2DF2 69              LD      L,C
2DF3 1600            LD      D,00H        ;DE = Offset zum nächsten
2DF5 5F              LD      E,A          ;Formatstring (Formatzeichen für
                                          ;den nächsten auszugebenden Wert)
2DF6 19              ADD     HL,DE        ;Offset addieren. HL zeigt auf
                                          ;die nächsten Formatangaben
2DF7 78              LD      A,B          ;A = Restlänge
2DF8 B7              OR      A            ;War die Restlänge = 0 ?
2DF9 C2032D          JP      NZ,2D03H     ;Nein: Formatstring ab (HL)
                                          ;verarbeiten
2DFC 1806            JR      2E04H        ;Ja: alten Formatstring auch für
                                          ;den nächsten Wert benutzen

; Stringende gefunden

2DFE CD492E          CALL    2E49H        ;'+' ausgeben wenn D <> 0
2E01 CD2A03          CALL    032AH        ;letztes Zeichen ausgeben
2E04 E1              POP     HL           ;PTZ zurück
2E05 F1              POP     AF           ;Flag bzw. nächstes Zeichen
                                          ;zurück (siehe 2DEAH)
2E06 C2CB2C          JP      NZ,2CCBH     ;Nächste Zahl mit gleichem
                                          ;Formatstring verarbeiten
2E09 DCFE20          CALL    C,20FEH      ;PRINT abschließen wenn das
                                          ;Anweisungsende erreicht wurde
                                          ;(siehe 2DD6H)
2E0C E3              EX      (SP),HL      ;PTZ retten, Vektoradresse zurück
2E0D CDDD29          CALL    29DDH        ;Formatstring aus der String-
                                          ;tabelle und dem Stringspeicher
                                          ;wieder löschen
2E10 E1              POP     HL           ;PTZ zurück
2E11 C36921          JP      2169H        ;Ausgaben wieder zum Bildschirm

; '!' gefunden

2E14 0E01            LD      C,01H        ;1 Textzeichen ausgeben
2E16 3EF1            LD      A,0F1H       ;--
```

```
: 2. '%'-Zeichen gefunden
: C = Anzahl der auszugebenden Textzeichen

*2E17   F1          POP    AF           :Stringzeiger aus dem Stack
                                        :nehmen
 2E18 05            DEC    B            :Zähler -1
 2E19 CD492E        CALL   2E49H        :'+' ausgeben wenn D <> 0
 2E1C E1            POP    HL           :PTZ zurück
 2E1D F1            POP    AF           :Flag zurück. Fertig ?
 2E1E 28E9          JR     Z,2E09H      :Ja: Print abschließen
 2E20 C5            PUSH   BC           :Nein: Stringzähler retten
 2E21 CD3723        CALL   2337H        :Auszugebenden String holen
 2E24 CDF40A        CALL   0AF4H        :TM-Error wenn kein String
 2E27 C1            POP    BC           :Stringzähler zurück
 2E28 C5            PUSH   BC           :Stringzähler retten
 2E29 E5            PUSH   HL  -        :PTZ retten
 2E2A 2A2141        LD     HL,(4121H)   :HL : Vektor des auszugebenden
                                        :Strings
 2E2D 41            LD     B,C          :B = Anzahl der auszugebenden
                                        :Zeichen
 2E2E 0E00          LD     C,00H        :C = 0: Ausgabe wird am String-
                                        :anfang begonnen
 2E30 C5            PUSH   BC           :Zähler retten
 2E31 CD682A        CALL   2A68H        :Auszugebenden Zeichen über LEFT$
                                        :holen
 2E34 CDAA28        CALL   28AAH        :Zeichen ausgeben
 2E37 2A2141        LD     HL,(4121H)   :HL : Vektor des ausgegebenen
                                        :Strings
 2E3A F1            POP    AF           :A = Anzahl der ausgegebenen
                                        :Zeichen
 2E3B 96            SUB    (HL)         :- Gesamtlänge des Strings
 2E3C 47            LD     B,A          :B = Restlänge des Strings
 2E3D 3E20          LD     A,20H        :A = Leerzeichen
 2E3F 04            INC    B            :Zähler +1
 2E40 05            DEC    B            :Zähler = 0 ?
 2E41 CAD32D        JP     Z,2DD3H      :Ja: nächstes Argument bearbeiten
 2E44 CD2A03        CALL   032AH        :Nein: Fehlende Zeichen durch
                                        :Leerzeichen ersetzen
 2E47 18F7          JR     2E40H        :Nächstes Zeichen

: '+' ausgeben wenn D <> 0

 2E49 F5            PUSH   AF           :AF retten
 2E4A 7A            LD     A,D          :D <> 0 ?
 2E4B B7            OR     A
 2E4C 3E2B          LD     A,2BH        :A = '+'
 2E4E C42A03        CALL   NZ,032AH     :Ja: '+' ausgeben
 2E51 F1            POP    AF           :AF zurück
 2E52 C9            RET
```

```
: EDIT-Ansprung nach SN-Error
: I: A = 00H

  2E53 329A40          LD      (409AH),A      :Letzen Fehlercode löschen
  2E56 2AEA40          LD      HL,(40EAH)     :HL = ERL
  2E59 B4              OR      H              :ERL = 65535 ?
  2E5A A5              AND     L              :War der SN-Error im aktiven
  2E5B 3C              INC     A              :Befehlsmodus ?
  2E5C EB              EX      DE,HL          :DE = ERL
  2E5D C8              RET     Z              :Ja: Kein EDIT ausführen
  2E5E 1804            JR      2E64H          :Nein: EDIT mit ZN = DE ausführen

: EDIT

  2E60 CD4F1E          CALL    1E4FH          :Zeilennummer nach DE holen
  2E63 C0              RET     NZ             :SN-Error ?
  2E64 E1              POP     HL             :RET-Adr aus dem Stack nehmen
  2E65 EB              EX      DE,HL          :HL = Zeilennummer
  2E66 22EC40          LD      (40ECH),HL     :als '.'-Zeile abspeichern
  2E69 EB              EX      DE,HL          :DE = Zeilennummer
  2E6A CD2C1B          CALL    1B2CH          :Zeile DE suchen
  2E6D D2D91E          JP      NC,1ED9H       :UL-Error wenn die Zeile nicht
                                              :existiert
  2E70 60              LD      H,B            :HL = Zeilenpointer
  2E71 69              LD      L,C
  2E72 23              INC     HL             :HL +2
  2E73 23              INC     HL
  2E74 4E              LD      C,(HL)         :BC = Zeilennummer
  2E75 23              INC     HL
  2E76 46              LD      B,(HL)
  2E77 23              INC     HL
  2E78 C5              PUSH    BC             :ZN retten
  2E79 CD7E2B          CALL    2B7EH          :Zeile ab (HL) decodieren und im
                                              :Zeilenbuffer ablegen
  2E7C E1              POP     HL             :ZN zurück
  2E7D E5              PUSH    HL             :und wieder retten
  2E7E CDAF0F          CALL    0FAFH          :Zeilennummer ausgeben
  2E81 3E20            LD      A,20H          :und Leerzeichen
  2E83 CD2A03          CALL    032AH          :dahinter
  2E86 2AA740          LD      HL,(40A7H)     :HL = Adresse des Zeilenbuffers
                                              :(= Zeiger auf den Zeilentext)
  2E89 3E0E            LD      A,0EH          :Cursor 'ein'
  2E8B CD2A03          CALL    032AH
  2E8E E5              PUSH    HL             :Zeiger retten
  2E8F 0EFF            LD      C,0FFH         :C = Zähler (auf -1 gesetzt wegen
                                              :INC C danach)
  2E91 0C              INC     C              :Zähler +1
  2E92 7E              LD      A,(HL)         :A = nächstes Zeichen
  2E93 B7              OR      A              :Zeilenende erreicht ?
  2E94 23              INC     HL             :Zeiger +1
  2E95 20FA            JR      NZ,2E91H       :Nein: weiterzählen
  2E97 E1              POP     HL             :Zeiger zurück
  2E98 47              LD      B,A            :B = 00H (Anzahl der bereits
                                              :ausgegebenen Zeichen)
                                              :C = Länge der Zeile
```

```
2E99 1600      LD    D,00H      ;D = 0 (Wiederholungszähler)
2E9B CD8403    CALL  0384H      ;Zeichen von der Tastatur holen
2E9E D630      SUB   30H        ;Ist es eine Ziffer ?
2EA0 380E      JR    C,2EB0H    ;Nein: weiter bei 2EB0H
2EA2 FE0A      CP    0AH        ;Ziffer ?
2EA4 300A      JR    NC,2EB0H   ;Nein: weiter bei 2EB0H
2EA6 5F        LD    E,A        ;Ja: E = Ziffernwert ( 00H - 09H)
2EA7 7A        LD    A,D        ;A = letzter Wert
2EA8 07        RLCA             ;*2
2EA9 07        RLCA             ;*2 (*4 gesamt)
2EAA 82        ADD   A,D        ;+Wert (*5 gesamt)
2EAB 07        RLCA             ;*2 (*10 gesamt)
2EAC 83        ADD   A,E        ;Nächste Dezimalstelle auf-
                                ;addieren
2EAD 57        LD    D,A        ;D = Zähler
2EAE 18EB      JR    2E9BH      ;Nächstes Zeichen holen

: Keine Ziffer eingegeben
: D = Wiederholungszähler
: A = ASCII-Code des eingegebenen Zeichens - 30H (!)

2EB0 E5        PUSH  HL         ;Zeiger retten
2EB1 21992E    LD    HL,2E99H   ;RET auf 2E99H setzen
2EB4 E3        EX    (SP),HL    ;Zeiger zurück
2EB5 15        DEC   D          ;Wiederholung gewünscht ?
2EB6 14        INC   D          ;Ist D <> 0 ?
2EB7 C2BB2E    JP    NZ,2EBBH   ;Ja: D belassen
2EBA 14        INC   D          ;Nein: D auf 1 setzen (gewünschte
                                ;Funktion einmal ausführen)
2EBB FED8      CP    0D8H       ;08H = Backspace (Linkspfeil) ?
2EBD CAD22F    JP    Z,2FD2H    ;Ja: weiter bei 2FD2H
2EC0 FEDD      CP    0DDH       ;0DH = RETURN ?
2EC2 CAE02F    JP    Z,2FE0H    ;Ja: weiter bei 2FE0H
2EC5 FEF0      CP    0F0H       ;20H = Leertaste ?
2EC7 2841      JR    Z,2F0AH    ;Ja: weiter bei 2F0AH
2EC9 FE31      CP    31H        ;Kleinbuchstabe ?
                                ;(Zeichen > 60H ?)
2ECB 3802      JR    C,2ECFH    ;Nein: Zeichen ok
2ECD D620      SUB   20H        ;Ja: In Großbuchstabe umwandeln
2ECF FE21      CP    21H        ;51H = 'Q' ?
2ED1 CAF62F    JP    Z,2FF6H    ;Ja: weiter bei 2FF6H
2ED4 FE1C      CP    1CH        ;4CH = 'L' ?
2ED6 CA402F    JP    Z,2F40H    ;Ja: weiter bei 2F40H
2ED9 FE23      CP    23H        ;53H = 'S' ?
2EDB 283F      JR    Z,2F1CH    ;Ja: weiter bei 2F1CH
2EDD FE19      CP    19H        ;49H = 'I' ?
2EDF CA7D2F    JP    Z,2F7DH    ;Ja: weiter bei 2F7DH
2EE2 FE14      CP    14H        ;44H = 'D' ?
2EE4 CA4A2F    JP    Z,2F4AH    ;Ja: weiter bei 2F4AH
2EE7 FE13      CP    13H        ;43H = 'C' ?
2EE9 CA652F    JP    Z,2F65H    ;Ja: weiter bei 2F65H
```

```
2EEC FE15          CP     15H         ;45H = 'E' ?
2EEE CAE32F        JP     Z,2FE3H     ;Ja: weiter bei 2FE3H
2EF1 FE28          CP     28H         ;58H = 'X' ?
2EF3 CA782F        JP     Z,2F78H     ;Ja: weiter bei 2F78H
2EF6 FE1B          CP     1BH         ;4BH = 'K' ?
2EF8 281C          JR     Z,2F16H     ;Ja: weiter bei 2F16H
2EFA FE18          CP     18H         ;48H = 'H' ?
2EFC CA752F        JP     Z,2F75H     ;Ja: weiter bei 2F75H
2EFF FE11          CP     11H         ;41H = 'A' ?
2F01 C0            RET    NZ          ;Nein: Rücksprung nach 2E99H

; A: Neu beginnen

2F02 C1            POP    BC          ;RET-Adr löschen
2F03 D1            POP    DE          ;ZN zurück
2F04 CDFE20        CALL   20FEH       ;Neue Bildschirmzeile beginnen
2F07 C3652E        JP     2E65H       ;EDIT neu beginnen

; Leertaste: Nächstes Zeichen anzeigen

2F0A 7E            LD     A,(HL)      ;Nächstes Zeichen holen
2F0B B7            OR     A           ;Zeilenende erreicht ?
2F0C C8            RET    Z           ;Ja: Fertig
2F0D 04            INC    B           ;Zähler +1
2F0E CD2A03        CALL   032AH       ;Zeichen ausgeben
2F11 23            INC    HL          ;Zeiger +1
2F12 15            DEC    D           ;Wiederholung ?
2F13 20F5          JR     NZ,2F0AH    ;Ja: nächstes Zeichen
2F15 C9            RET                ;Nein: Fertig

; K: Zeile bis zum eingegebenen Zeichen löschen

2F16 E5            PUSH   HL          ;Zeiger retten
2F17 215F2F        LD     HL,2F5FH    ;RET-Adr auf 2F5FH setzen (Zur
                                      ;Ausgabe von '!' am Ende)
2F1A E3            EX     (SP),HL     ;Zeiger zurück
2F1B 37            SCF                ;CY = 1 (Kennung für Zeichen
                                      ;löschen)

; S: Eingegebenes Zeichen suchen

2F1C F5            PUSH   AF          ;Flags retten
2F1D CD8403        CALL   0384H       ;Zeichen holen
2F20 5F            LD     E,A         ;E = Zeichen bis zu dem gelöscht
                                      ;werden soll
2F21 F1            POP    AF          ;Flags zurück
2F22 F5            PUSH   AF          ;und wieder retten
2F23 DC5F2F        CALL   C,2F5FH     ;'!' ausgeben für 'K'-Funktion
2F26 7E            LD     A,(HL)      ;A = nächstes Zeichen
2F27 B7            OR     A           ;Zeilenende erreicht ?
2F28 CA3E2F        JP     Z,2F3EH     ;Ja: Fertig
2F2B CD2A03        CALL   032AH       ;Nein: Zeichen ausgeben
```

```
2F2E F1           POP    AF          :Flags zurück
2F2F F5           PUSH   AF          :und wieder retten
2F30 DCA12F       CALL   C,2FA1H     :Zeichen löschen für 'K'-Funktion
2F33 3802         JR     C,2F37H     :Nächste Befehle bei 'K'
                                     :überspringen (Zähler und Zeiger
                                     :wurden in 2FA1H angepasst)
2F35 23           INC    HL          :Zeiger +1
2F36 04           INC    B           :Zähler +1
2F37 7E           LD     A,(HL)      :A = nächstes Zeichen
2F38 BB           CP     E           := gesuchtem Zeichen ?
2F39 20EB         JR     NZ,2F26H    :Nein: nächstes Zeichen
2F3B 15           DEC    D           :Ja: D-tes Zeichen gefunden ?
2F3C 20E8         JR     NZ,2F26H    :Nein: nächstes Zeichen
2F3E F1           POP    AF          :Ja: Flags zurück
2F3F C9           RET

: L: Zeile listen und Edit neu beginnen

2F40 CD752B       CALL   2B75H       :Zeile ausgeben
2F43 CDFE20       CALL   20FEH       :Neue Bildschirmzeile beginnen
2F46 C1           POP    BC          :RET-Adr löschen
2F47 C37C2E       JP     2E7CH       :Edit neu beginnen

: D: Zeichen löschen

2F4A 7E           LD     A,(HL)      :A = Zeichen
2F4B B7           OR     A           :Zeilenende erreicht ?
2F4C C8           RET    Z           :Ja: Fertig
2F4D 3E21         LD     A,21H       :Nein: '!' ausgeben
2F4F CD2A03       CALL   032AH
2F52 7E           LD     A,(HL)      :A = Zeichen
2F53 B7           OR     A           :Zeilenende ?
2F54 2809         JR     Z,2F5FH     :Ja: Fertig
2F56 CD2A03       CALL   032AH       :Nein: Zeichen ausgeben
2F59 CDA12F       CALL   2FA1H       :und löschen
2F5C 15           DEC    D           :Wiederholung ?
2F5D 20F3         JR     NZ,2F52H    :Ja: nächstes Zeichen

: '!' ausgeben

2F5F 3E21         LD     A,21H       :A = '!'-Zeichen
2F61 CD2A03       CALL   032AH       :Zeichen ausgeben
2F64 C9           RET

: C: Zeichen verändern

2F65 7E           LD     A,(HL)      :A = Zeichen
2F66 B7           OR     A           :Zeilenende erreicht ?
2F67 C8           RET    Z           :Ja: Fertig
2F68 CD8403       CALL   0384H       :Nein: neues Zeichen holen
2F6B 77           LD     (HL),A      :einsetzen
2F6C CD2A03       CALL   032AH       :und ausgeben
2F6F 23           INC    HL          :Zeiger +1
2F70 04           INC    B           :Zähler +1
2F71 15           DEC    D           :Wiederholung ?
2F72 20F1         JR     NZ,2F65H    :Ja: nächstes Zeichen
2F74 C9           RET                :Nein: Fertig
```

```
; H: Rest der Zeile löschen und nach 'I' springen

  2F75 3600         LD    (HL),00H    :Zeilenende an die aktuelle
                                      :Position setzen
  2F77 48           LD    C,B         :C = Anzahl der bereits
                                      :ausgegebenen Zeichen = neue
                                      :Zeilenlänge

; X: Rest der Zeile ausgeben und nach 'I' springen

  2F78 16FF         LD    D,0FFH      :Zur Ausgabe des Rests der Zeile
  2F7A CD0A2F       CALL  2F0AH       :255 mal die Leertastenfunktion
                                      :ausführen

; I: Neue Zeichen einsetzen

  2F7D CD8403       CALL  0384H       :Neues Zeichen holen
  2F80 B7           OR    A           :(warum ?)
  2F81 CA7D2F       JP    Z,2F7DH     :(siehe 0384H)
  2F84 FE08         CP    08H         :Linkspfeil ?
  2F86 280A         JR    Z,2F92H     :Ja: Zeichen löschen
  2F88 FE0D         CP    0DH         :RETURN ?
  2F8A CAE02F       JP    Z,2FE0H     :Ja: RETURN-Funktion ausführen
  2F8D FE1B         CP    1BH         :Shift-Hochpfeil (I-Funktion
                                      :beenden) ?
  2F8F C8           RET   Z           :Ja: Fertig
  2F90 201E         JR    NZ,2FB0H    :Nein: Zeichen einfügen

; Linkspfeil (Zeichen löschen) in der I-Funktion

  2F92 3E08         LD    A,08H       :A = ASCII-Code für Zeichen
                                      :löschen (Backspace)
  2F94 05           DEC   B           :Schon Zeichen ausgegeben ?
  2F95 04           INC   B           :(Sind noch Zeichen zum Löschen
                                      :vorhanden ?)
  2F96 281F         JR    Z,2FB7H     :Nein: Fertig
  2F98 CD2A03       CALL  032AH       :Ja: Zeichen auf dem Bildschirm
                                      :löschen
  2F9B 2B           DEC   HL          :Zeiger -1
  2F9C 05           DEC   B           :Zähler -1
  2F9D 117D2F       LD    DE,2F7DH    :RET-Adr auf 2FD7H setzen
  2FA0 D5           PUSH  DE

; Zeichen bei (HL) löschen

  2FA1 E5           PUSH  HL          :Zeiger retten
  2FA2 0D           DEC   C           :Zeilenlänge -1
  2FA3 7E           LD    A,(HL)      :Zeichen holen
  2FA4 B7           OR    A           :Zeilenende erreicht ?
  2FA5 37           SCF               :CY = 1
  2FA6 CA9008       JP    Z,0890H     :Ja: Zeiger zurück, Fertig
  2FA9 23           INC   HL          :Nein: Zeiger +1
  2FAA 7E           LD    A,(HL)      :nächstes Zeichen holen
  2FAB 2B           DEC   HL          :Zeiger -1
  2FAC 77           LD    (HL),A      :und an aktueller Stelle
                                      :einsetzen
```

```
2FAD 23             INC    HL          ;Zeiger +1
2FAE 18F3           JR     2FA3H       ;Zeile weiter verschieben bis das
                                       ;Zeilenende erreicht wird
```

```
; Ein Zeichen bei (HL) einfügen

2FB0 F5             PUSH   AF          ;Zeichen retten
2FB1 79             LD     A,C         ;A = Zeilenlänge
2FB2 FEFF           CP     0FFH        ;Maximale Länge schon erreicht ?
2FB4 3803           JR     C,2FB9H     ;Nein: Zeichen einfügen
2FB6 F1             POP    AF          ;Ja: Zeichen zurück
2FB7 18C4           JR     2F7DH       ;zurück zur 'I'-Funktion
```

```
; Zeichen einfügen

2FB9 90             SUB    B           ;A = Zeilenlänge - Anzahl der
                                       ;schon ausgegebenen Zeichen
                                       ;= Restlänge der Zeile
2FBA 0C             INC    C           ;Zeilenlänge +1
2FBB 04             INC    B           ;Ausgegebene Zeichen +1
2FBC C5             PUSH   BC          ;Zähler retten
2FBD EB             EX     DE,HL       ;DE = Zeiger
2FBE 6F             LD     L,A         ;HL = Restlänge der Zeile
2FBF 2600           LD     H,00H
2FC1 19             ADD    HL,DE       ;HL = aktueller Zeiger + Rest-
                                       ;länge = Zeiger zum Ende der
                                       ;Zeile
2FC2 44             LD     B,H         ;BC = Endzeiger
2FC3 4D             LD     C,L
2FC4 23             INC    HL          ;HL = Endzeiger +1
2FC5 CD5819         CALL   1958H       ;Zeile verschieben
2FC8 C1             POP    BC          ;Zähler zurück
2FC9 F1             POP    AF          ;Zeichen zurück
2FCA 77             LD     (HL),A      ;Zeichen einsetzen
2FCB CD2A03         CALL   032AH       ;und ausgeben
2FCE 23             INC    HL          ;Zeiger +1
2FCF C37D2F         JP     2F7DH       ;zurück zur 'I'-Funktion
```

```
; Linkspfeil: Zeichen links vom Cursor löschen

2FD2 78             LD     A,B         ;A = Anzahl der ausgegebenen
                                       ;Zeichen (= Anzahl der Zeichen
                                       ;links vom Cursor)
2FD3 B7             OR     A           ;Schon Zeichen ausgegeben ?
2FD4 C8             RET    Z           ;Nein: Fertig
2FD5 05             DEC    B           ;Ja: Zähler -1
2FD6 2B             DEC    HL          ;Zeiger -1
2FD7 3E08           LD     A,08H       ;Zeichen auf dem Bildschirm
2FD9 CD2A03         CALL   032AH       ;löschen
2FDC 15             DEC    D           ;Wiederholung ?
2FDD 20F3           JR     NZ,2FD2H    ;Ja: nächstes Zeichen
2FDF C9             RET
```

```
; RETURN: Rest der Zeile ausgeben und Edit beenden

  2FE0 CD752B            CALL    2B75H          :Rest der Zeile ausgeben

; E: Edit beenden

  2FE3 CDFE20            CALL    20FEH          :Neue Bildschirmzeile beginnen
  2FE6 C1                POP     BC             :RET-Adr löschen
  2FE7 D1                POP     DE             :ZN zurück
  2FE8 7A                LD      A,D            :ZN = 65535 ?
  2FE9 A3                AND     E              :(Z setzen für 2FEFH,
  2FEA 3C                INC     A              :siehe 1A71H ff)
  2FEB 2AA740            LD      HL,(40A7H)     :HL = Adresse des Zeilenbuffers
  2FEE 2B                DEC     HL             :HL -1
  2FEF C8                RET     Z              :Fertig wenn ZN = 65535
  2FF0 37                SCF                    :CY = 1
  2FF1 23                INC     HL             :Zeiger +1
  2FF2 F5                PUSH    AF             :Flags retten
  2FF3 C3981A            JP      1A98H          :Zeile ab HL wieder in den
                                                :Programmtext übernehmen

; Q: Edit beenden ohne die Änderungen zu übernehmen

  2FF6 C1                POP     BC             :RET-Adr zurück
  2FF7 D1                POP     DE             :ZN zurück
  2FF8 C3191A            JP      1A19H          :Zurück zum aktiven Befehls-
                                                :modus

  2FFB DEC3              SBC     A,0C3H         :--
  2FFD C344B2            JP      0B244H
```

```
: Tastatureingabe mit FKEY-Überprüfung
: (wird nur bei der Zeileneingabe (05D9H ff) verwendet und
; hat daher als Rücksprungadresse 05E3H ! )
: I: HL = Bufferzeiger

  3000 C31534          JP     3415H        ;weiter bei 3415H

: Keine Funktionstaste gedrückt (Ansprung von 3458H ff)
: Wurde CTRL und eine Ziffer von 1 bis 8 zur Farbumschaltung gedrückt ?

  3003 E5              PUSH   HL           :Zeiger retten
  3004 211840          LD     HL,4018H     ;HL : CTRL-Byte
  3007 CB7E            BIT    07H,(HL)     ;wurde beim letzten Mal CTRL
                                           :gedrückt ?
  3009 2812            JR     Z,301DH      :Nein: Fertig
  300B CBBE            RES    07H,(HL)     :Ja: Bit wieder löschen
  300D FE31            CP     31H          ;wurde jetzt eine Ziffer
                                           :gedrückt ?
  300F 380C            JR     C,301DH      :Nein: Zeichen in A übergeben
  3011 FE39            CP     39H          :Ziffer ?
  3013 3008            JR     NC,301DH     :Nein: Fertig
  3015 D631            SUB    31H          :Ja: A = Colourcode 0 - 7
  3017 CD2136          CALL   3621H        :Neuen Colourcode abspeichern und
                                           ;Cursor einschalten
  301A E1              POP    HL           :Zeiger zurück
  301B 18E3            JR     3000H        :neues Zeichen holen

: Zeichen in A an Eingaberoutine übergeben

  301D E1              POP    HL           :Zeiger zurück
  301E C3E305          JP     05E3H        :Rücksprung zur Zeileneingabe

  3021 FF              RST    38H          ;--

: Grafikzeichen ( > 7FH) erkannt

  3022 FECO            CP     0C0H         :Dieser Vergleich diente in einer
                                           :früheren Version dazu, die
                                           :Zeichen von 192 - 255 (C0H -
                                           :FFH) als Tabulatorfunktion
                                           :(Ausgabe von 0 bis 63 Leer-
                                           :zeichen) zu verwenden. In der
                                           :jetzigen Version werden aber
                                           :alle Werte > 7FH als Grafik-
                                           :zeichen dargestellt
  3024 C30531          JP     3105H        :Zeichen ausgeben
```

```
; Ehemalige Tabulatorfunktion (unbenutzt)

  3027 D6C0          SUB    0C0H        ;A = Wert - 192 (= Anzahl der
                                        ;auszugebenden Leerzeichen)
  3029 CA0831        JP     Z,3108H     ;A = 0, Ja: Fertig
  302C 47            LD     B,A         ;Nein: B = Zähler
  302D 3E20          LD     A,20H       ;A = Leerzeichen
  302F C5            PUSH   BC          ;Zähler retten
  3030 CD8431        CALL   3184H       ;Zeichen ausgeben
  3033 C1            POP    BC          ;Zähler zurück
  3034 10F7          DJNZ   302DH       ;nächstes Zeichen
  3036 C30831        JP     3108H

; Cursor ein Zeichen nach rechts setzen ( = CHR$(25) )

  3039 23            INC    HL          ;Adresse +1
  303A E5            PUSH   HL          ;Adresse retten
  303B CD6531        CALL   3165H       ;HL = Startadresse der aktuellen
                                        ;Zeile
  303E EB            EX     DE,HL       ;DE = HL
  303F E1            POP    HL          ;Neue Cursoradresse zurück
  3040 DF            RST    18H         ;Wurde die nächste Zeile
                                        ;erreicht ? (steht der Cursor
                                        ;jetzt am Zeilenanfang ?)
  3041 C0            RET    NZ          ;Nein: ok
  3042 11D8FF        LD     DE,0FFD8H   ;Ja: DE = -40
  3045 19            ADD    HL,DE       ;40 (Eine Zeilenlänge) von der
                                        ;neuen Adresse abziehen: Der
                                        ;Cursor steht wieder am Anfang
                                        ;der alten Zeile
  3046 C9            RET

; CRTC programmieren
; I: A  = Register des CRTC
;    H  = 1. Wert (wird im Register A abgespeichert)
;    L  = 2. Wert (wird im Register A+1 abgespeichert)

  3047 C5            PUSH   BC          ;Register retten
  3048 E5            PUSH   HL
  3049 0602          LD     B,02H       ;2 Durchgänge
  304B 0EFA          LD     C,0FAH      ;C = Portadresse zur CRTC Anwahl
  304D ED79          OUT    (C),A       ;Register wählen
  304F 0C            INC    C           ;C = Portadresse +1
  3050 ED61          OUT    (C),H       ;Wert übergeben
  3052 3C            INC    A           ;A +1: Beim nächsten Durchgang
                                        ;das nächste Register anwählen
  3053 65            LD     H,L         ;und den Wert in L übergeben
  3054 10F5          DJNZ   304BH       ;Nächster Durchgang
  3056 E1            POP    HL          ;Register zurück
  3057 C1            POP    BC
  3058 C9            RET
```

```
; Cursor 'aus' ( =CHR$(15) )

  3059 E5               PUSH   HL            :Adresse retten
  305A 210720           LD     HL.2007H      :H und L = Werte zur CRTC-
                                             :Programmierung (siehe S. 114 im
                                             :Handbuch)
  305D 1804             JR     3063H         :weiter bei 3063H

; Cursor 'ein' ( =CHR$(14) )

  305F E5               PUSH   HL            :Adresse retten
  3060 2A1940           LD     HL.(4019H)    :H und L = Werte zur CRTC-
                                             :programmierung
  3063 3E0A             LD     A.0AH         :Register 10 und 11 anwählen
  3065 CD4730           CALL   3047H         :CRTC programmieren
  3068 7C               LD     A.H -         :A = 1. Wert
  3069 E1               POP    HL            :Adresse zurück
  306A E5               PUSH   HL            :Register retten
  306B D5               PUSH   DE
  306C FE20             CP     20H           :Wurde der Cursor ausgeschaltet ?
  306E C41436           CALL   NZ.3614H      :Nein: Farbe des neuen Cursors
                                             :setzen
  3071 00               NOP                  :--
  3072 3E0E             LD     A.0EH         :Register 14 und 15 mit der neuen
  3074 CD4730           CALL   3047H         :Adresse (in HL) programmieren
  3077 D1               POP    DE            :Register zurück
  3078 E1               POP    HL
  3079 C9               RET

; Neuen Farbcode in den Farbspeicher setzen
; I: HL = Bildschirmadresse
;     A  = Neuer Farbcode

  307A E5               PUSH   HL            :Register retten
  307B D5               PUSH   DE
  307C F5               PUSH   AF
  307D C3F835           JP     35F8H         :weiter bei 35F8H

; Neuen Wert im Farbspeicher programmieren (siehe 35F8H ff)

  3080 19               ADD    HL.DE         :Adresse im Farbspeicher
                                             :errechnen
  3081 E5               PUSH   HL            :Adresse retten
  3082 219043           LD     HL.4390H      :HL = Adresse der Farbcodetabelle
  3085 110000           LD     DE.0000H      :DE = Offset für Tabelle
  3088 3A2340           LD     A.(4023H)     :A = Farbcode
  308B 5F               LD     E.A           :DE = Offset
  308C 19               ADD    HL.DE         :HL = Adresse + Offset
  308D 7E               LD     A.(HL)        :Echten Farbcode holen
  308E E1               POP    HL            :Farbspeicheradresse zurück
  308F 77               LD     (HL).A        :Neue Farbe setzen
  3090 F1               POP    AF            :Register zurück (von 307AH)
  3091 D1               POP    DE
  3092 E1               POP    HL
  3093 C9               RET
```

```
3094 03                                         :--
3095 01
3096 02
3097 04
3098 06
3099 08
309A 09
309B 0A
309C 05

; Neuen POS-Wert errechnen (AF)
; I: --
; O: A = Neuer POS-Wert

 309D E5          PUSH   HL          ;Register retten
 309E D5          PUSH   DE
 309F C5          PUSH   BC
 30A0 2A2040      LD     HL,(4020H)  ;HL = Neue Cursoradresse
 30A3 E5          PUSH   HL          ;Adresse retten
 30A4 CD6531      CALL   3165H       ;Zeilenanfang errechnen
 30A7 EB          EX     DE,HL       ;DE = Adresse des Zeilenanfangs
 30A8 E1          POP    HL          ;HL = Cursoradresse
 30A9 B7          OR     A           ;CY = 0
 30AA ED52        SBC    HL,DE       ;HL = Cursoradresse - Zeilen-
                                     ;anfangsadresse = Position des
                                     ;Cursors in der Zeile
 30AC 7D          LD     A,L         ;A = POS-Wert
 30AD C1          POP    BC          ;Register zurück
 30AE D1          POP    DE
 30AF E1          POP    HL
 30B0 C9          RET

; Tabulatorwert prüfen
; I: E = Gewünschter Tabulatorwert
; O: A = Korrekter Tabulatorwert
;    E = A

 30B1 7B          LD     A,E         ;A = TAB-Wert
 30B2 5F          LD     E,A         ;E = TAB-Wert
 30B3 3A9C40      LD     A,(409CH)   ;A = Ausgabeflag
 30B6 B7          OR     A           ;Cassettenausgabe ?
 30B7 FA4A1E      JP     M,1E4AH     ;Ja: FC-Error
 30BA 2805        JR     Z,30C1H     ;Sprung bei Bildschirm-
                                     ;ausgabe
 30BC 3A9E40      LD     A,(409EH)   ;Druckerausgabe: A = höchste
                                     ;erreichbare Tabulatorposition
 30BF 1803        JR     30C4H       ;weiter bei 30C4H
```

```
; Bildschirmausgabe

  30C1 3A9D40        LD     A,(409DH)   :A = Anzahl der Zeichen pro Zeile
  30C4 BB            CP     E           :Ist der gewünschte TAB-Wert
                                        :größer als die höchtste
                                        :TAB-Position bzw größer als die
                                        :Zeilenlänge
  30C5 300B          JR     NC,30D2H    :Nein: Wert ok
  30C7 C5            PUSH   BC          :Ja: BC retten
  30C8 43            LD     B,E         :A und E vertauschen
  30C9 5F            LD     E,A
  30CA 78            LD     A,B
  30CB C1            POP    BC          :BC zurück
  30CC 93            SUB    E           :Zeilenlänge bzw höchsten
                                        :TAB-Wert abziehen
  30CD 30FD          JR     NC,30CCH    :weiter wenn Rest noch > 0
  30CF 83            ADD    A,E         :Letzte Subtraktion rückgängig
                                        :machen
  30D0 5F            LD     E,A         :E = Neuer TAB-Wert (nie größer
                                        :als die Zeilenlänge)
  30D1 C9            RET

; TAB-Wert in Ordnung

  30D2 7B            LD     A,E         :Wert nach A übergeben
  30D3 C9            RET

; UPRO für PRINT § (siehe 207AH)
; I: DE = § - Argument

  30D4 21E703        LD     HL,03E7H    :HL = 999
  30D7 DF            RST    18H         :Ist der §-Wert größer als 999 ?
  30D8 E1            POP    HL          :PTZ zurück
  30D9 DA4A1E        JP     C,1E4AH     :Ja: FC-Error
  30DC C37E20        JP     207EH       :Nein: Fertig

; Neuen POS-Wert errechnen (wie 309DH aber Wert in E und A)

  30DF CD9D30        CALL   309DH       :POS errechnen
  30E2 5F            LD     E,A         :und auch nach E schreiben
  30E3 C9            RET
```

```
: Bildschirmroutine (wird über DCB-Aufruf angesprochen)
: I: IX = Adresse des Bildschirm-DCBs (= 401DH)
:     C  = auszugebendes Zeichen

  30E4 DD6E03          LD     L,(IX+03H)   ;HL = aktuelle Cursoradresse
  30E7 DD6604          LD     H,(IX+04H)   ;(aus DCB)
  30EA DA7D31          JP     C,317DH      ;Falscher DCB-Typ ?
                                           ;Ja: weiter bei 317DH
  30ED DD7E05          LD     A,(IX+05H)   ;Cursor eingeschaltet ?
  30F0 B7              OR     A
  30F1 2005            JR     NZ,30F8H     ;Ja: weiter bei 30F8H
  30F3 CD5930          CALL   3059H        ;Nein: Cursor 'aus'
  30F6 1803            JR     30FBH        ;weiter bei 30FBH
  30F8 CD5F30          CALL   305FH        ;Cursor 'ein'
  30FB 79              LD     A,C          ;A = auszugebendes Zeichen
  30FC FE20            CP     20H          ;Steuercode ?
  30FE 3822            JR     C,3122H      ;Ja: weiter bei 3122H
  3100 FE80            CP     80H          ;Grafikzeichen ?
  3102 D22230          JP     NC,3022H     ;Ja: weiter bei 3022H
  3105 CD8431          CALL   3184H        ;Zeichen ausgeben
  3108 7E              LD     A,(HL)       ;A = Zeichen an der neuen Cursor-
                                           ;position
  3109 57              LD     D,A          ;D = Zeichen
  310A DD7E05          LD     A,(IX+05H)   ;Cursor eingeschaltet ?
  310D B7              OR     A
  310E 2808            JR     Z,3118H      ;Nein: weiter bei 3118H
  3110 DD7205          LD     (IX+05H),D   ;Ja: Zeichen abspeichern
  3113 CD5F30          CALL   305FH        ;Cursor 'ein'
  3116 1803            JR     311BH        ;weiter bei 311BH
  3118 CD5930          CALL   3059H        ;Cursor 'aus'
  311B DD7503          LD     (IX+03H),L   ;Neue Cursoradresse
  311E DD7404          LD     (IX+04H),H   ;abspeichern
  3121 C9              RET

: Steuercode ( Wert < 32) erkannt (siehe S. 120 im Handbuch)
: I: A  = Steuercode
:    HL = Cursoradresse

  3122 110831          LD     DE,3108H     ;RET-Adr auf 3108H setzen
  3125 D5              PUSH   DE
  3126 FE08            CP     08H          ;Backspace ?
  3128 CADF31          JP     Z,31DFH      ;Ja: weiter bei 31DFH
  312B FE0A            CP     0AH          ;Code < 10 ?
  312D D8              RET    C            ;Ja: Fertig
  312E FE0E            CP     0EH          ;Cursor 'ein' ?
  3130 DAC931          JP     C,31C9H      ;Sprung wenn Zeichen < 0EH
  3133 CAF831          JP     Z,31F8H      ;Ja: weiter bei 31F8H
  3136 FE0F            CP     0FH          ;Cursor 'aus'
  3138 CAFD31          JP     Z,31FDH      ;Ja: weiter bei 31FDH
  313B FE18            CP     18H          ;Zeichen < 24 ?
  313D D8              RET    C            ;Ja: Fertig
```

```
  313E FE18           CP      18H          ;Cursor nach links ?
  3140 CAE531         JP      Z,31E5H      ;Ja: weiter bei 31E5H
  3143 FE19           CP      19H          ;Cursor nach rechts ?
  3145 CA3930         JP      Z,3039H      ;Ja: weiter bei 3039H
  3148 FE1A           CP      1AH          ;Cursor nach unten ?
  314A CA0032         JP      Z,3200H      ;Ja: weiter bei 3200H
  314D FE1B           CP      1BH          ;Cursor nach oben ?
  314F CA1232         JP      Z,3212H      ;Ja: weiter bei 3212H
  3152 FE1C           CP      1CH          ;Cursor nach links-oben ?
  3154 CAD431         JP      Z,31D4H      ;Ja: weiter bei 31D4H
  3157 FE1D           CP      1DH          ;Cursor zum Zeilenanfang ?
  3159 CAD931         JP      Z,31D9H      ;Ja: weiter bei 31D9H
  315C FE1E           CP      1EH          ;Löschen bis zum Ende der Zeile ?
  315E 285F           JR      Z,31BFH      ;Ja: weiter bei 31BFH
  3160 FE1F           CP      1FH          ;Löschen bis zum Ende des Bild-
                                           ;schirms ?
  3162 2845           JR      Z,31A9H      ;Ja: weiter bei 31A9H
  3164 C9             RET

; Adresse des Zeilenanfangs der aktuellen Zeile errechnen
; I: HL = aktuelle Cursoradresse
; O: HL = Adresse des Zeilenanfangs

  3165 1100BC         LD      DE,0BC00H    ;DE = -4400H
  3168 0601           LD      B,01H        ;Zähler auf 1 setzen
  316A 19             ADD     HL,DE        ;HL = aktueller §-Wert
                                           ;(= Adresse - 4400H)
  316B 112800         LD      DE,0028H     ;DE = 40 (Zeilenlänge)
  316E B7             OR      A            ;CY = 0
  316F ED52           SBC     HL,DE        ;40 abziehen
  3171 3803           JR      C,3176H      ;Sprung wenn Rest < 0
  3173 04             INC     B            ;Zähler +1
  3174 18F5           JR      316BH        ;weiter
  3176 21D843         LD      HL,43D8H     ;HL = 4400H - 28
  3179 19             ADD     HL,DE        ;Zeilenlänge aufaddieren
  317A 10FD           DJNZ    3179H        ;B mal die Zeilenlänge addieren
  317C C9             RET

; Falschen DCB-Typ erkannt

  317D DD7E05         LD      A,(IX+05H)   ;Cursor eingeschaltet ?
  3180 B7             OR      A
  3181 C0             RET     NZ           ;Ja: ok
  3182 7E             LD      A,(HL)       ;Nein: A = Zeichen an der
                                           ;nächsten Bildschirmposition
  3183 C9             RET
```

```
: Zeichen auf dem Bildschirm darstellen
: I: HL = Bildschirmadresse
:     A  = Zeichen

  3184 CD7A30        CALL   307AH        :Farbspeicher setzen
  3187 77            LD     (HL),A       :Zeichen darstellen
  3188 23            INC    HL           :Adresse +1
  3189 11E847        LD     DE,47E8H     :DE = Adresse des Bildschirmendes
  318C DF            RST    18H          :Bildschirmende erreicht ?
  318D D8            RET    C            :Nein: ok

: Bildschirm um eine Zeile nach oben schieben (Scrolling)

  318E 21E847        LD     HL,47E8H     :HL = Endadresse des Bildschirms
  3191 11D8FF        LD     DE,0FFD8H    :DE = -40
  3194 19            ADD    HL,DE        :HL = HL - 40 = Startadresse der
                                         :letzten Bildschirmzeile
  3195 E5            PUSH   HL           :HL retten
  3196 CD3936        CALL   3639H        :Register vorbereiten
  3199 EDA0          LDI                 :Zeichenspeicher verschieben
  319B D9            EXX                 :Register tauschen
  319C EDA0          LDI                 :Farbspeicher verschieben
  319E D9            EXX                 :Register tauschen
  319F 78            LD     A,B          :Fertig ?
  31A0 B1            OR     C            :BC = 0 ?
  31A1 20F6          JR     NZ,3199H     :Nein: weiter verschieben
  31A3 D9            EXX                 :Register tauschen
  31A4 E1            POP    HL           :2. Registersatz zurück
  31A5 D1            POP    DE           :(siehe 3639H ff)
  31A6 C1            POP    BC
  31A7 D9            EXX
  31A8 E1            POP    HL           :Startadresse der letzten
                                         :Bildschirmzeile zurück

: Alle Zeichen von HL bis zum Ende des Bildschirmspeichers löschen
: ( = CHR$(31) )

  31A9 11E847        LD     DE,47E8H     :DE = Endadresse der letzten
                                         :Bildschirmzeile
  31AC E5            PUSH   HL           :HL retten
  31AD CDE031        CALL   31E0H        :Zeichen in (HL) löschen und
                                         :Farbe auf COLOUR setzen
  31B0 23            INC    HL           :Adresse +1
  31B1 DF            RST    18H          :Ende erreicht ?
  31B2 20F9          JR     NZ,31ADH     :Nein: weiter
  31B4 E1            POP    HL           :HL zurück
  31B5 C9            RET
```

```
; RENUM

  31B6 FDE5          PUSH   IY          :IY retten
  31B8 1868          JR     3222H       :weiter bei 3222H

; Rücksprung von RENUM

  31BA FDE1          POP    IY          :IY zurück
  31BC C3832C        JP     2C33H       :Alle Zeiger im Programm erneuern

; Alle Zeichen bis zum Ende der Zeile löschen ( = CHR$(30) )

  31BF E5            PUSH   HL          :Adresse retten
  31C0 CD6531        CALL   3165H       :HL = Zeilenanfangsadresse
  31C3 EB            EX     DE,HL       :DE = HL
  31C4 19            ADD    HL,DE       :HL = Zeilenendadresse + 1
                                        :(= DE + 40)
  31C5 EB            EX     DE,HL       :DE = Zeilenendadresse
  31C6 E1            POP    HL          :HL = Aktuelle Adresse
  31C7 18E3          JR     31ACH       :Alle Zeichen von HL bis DE
                                        :löschen

; Neue Zeile beginne ( = CHR$(13) )

  31C9 CD1B36        CALL   361BH       :Aktuelle Zeile löschen
  31CC 19            ADD    HL,DE       :HL = Startadresse der nächsten
                                        :Zeile
  31CD 11E847        LD     DE,47E8H    :DE = Endadresse des Bildschirms
  31D0 DF            RST    18H         :Ende erreicht ?
  31D1 28BE          JR     Z,3191H     :Ja: Scrollen
  31D3 C9            RET                :Nein: Fertig

; Cursor nach links oben ( = CHR$(28) )

  31D4 210044        LD     HL,4400H    :HL = Startadresse des Bild-
                                        :schirms
  31D7 1803          JR     31DCH       :als neue Adresse setzen

; Cursor zum Zeilenanfang ( = CHR$(29) )

  31D9 CD6531        CALL   3165H       :HL = Zeilenanfangsadresse
  31DC C35F30        JP     305FH       :Cursor 'ein'

; Letztes Zeichen löschen ( = CHR$(8) )

  31DF 2B            DEC    HL          :HL = Adresse -1
  31E0 3E20          LD     A,20H       :Letztes Zeichen löschen
  31E2 C3EE35        JP     35EEH
```

```
; Cursor ein Zeichen nach links ( = CHR$(24) )

   31E5 E5              PUSH    HL            :Adresse retten
   31E6 CD6531          CALL    3165H         :Zeilenanfangsadresse holen
   31E9 EB              EX      DE,HL         :DE = Zeilenanfang
   31EA E1              POP     HL            :Adresse zurück
   31EB DF              RST     18H           :Zeilenanfang erreicht ?
   31EC 2803            JR      Z,31F1H       :Ja: Cursor zum Zeilenende
                                              :(in der Zeile bleiben)
   31EE 2B              DEC     HL            :Nein: HL = Adresse -1
   31EF 18EB            JR      31DCH         :Cursor 'ein'

; Cursor zum Zeilenende

   31F1 2B              DEC     HL            :HL = Adresse -1
   31F2 112800          LD      DE,0028H      :+ Zeilenlänge = Endadresse
   31F5 19              ADD     HL,DE         :der Zeile
   31F6 18E4            JR      31DCH         :Cursor 'ein'

; Cursor 'ein' ( = CHR$(14) )

   31F8 7E              LD      A,(HL)        :A = Zeichen an der Cursor-
                                              :position (A <> 0 !)
   31F9 DD7705          LD      (IX+05H),A    :Neuen Wert abspeichern
   31FC C9              RET

; Cursor 'aus' ( = CHR$(15) )

   31FD AF              XOR     A             :A = 0
   31FE 18F9            JR      31F9H         :(IX+05H) auf 0 setzen

; Cursor eine Zeile tiefer ( = CHR$(26) )

   3200 CD0736          CALL    3607H         :Cursor löschen. DE = 40
   3203 19              ADD     HL,DE         :Eine Zeilenlänge addieren
   3204 B7              OR      A             :CY = 0
   3205 3F              CCF                   :CY = 1
   3206 11E847          LD      DE,47E8H      :Ende des Bildschirms
   3209 DF              RST     18H           :überschritten ?
   320A 3804            JR      C,3210H       :Nein: ok
   320C 1140FC          LD      DE,0FC40H     :Ja: DE = -960
   320F 19              ADD     HL,DE         :HL = HL - 960 ( 24 * 40 abziehen
                                              :damit der Cursor oben wieder
                                              :erscheint (wrap-around))
   3210 18CA            JR      31DCH         :Cursor 'ein'

; Cursor eine Zeile höher ( = CHR$(27) )

   3212 CD0F36          CALL    360FH         :Cursor löschen. DE = -40
   3215 19              ADD     HL,DE         :Eine Zeilenlänge abziehen
   3216 110044          LD      DE,4400H      :Anfang des Bildschirms
   3219 DF              RST     18H           :erreicht ?
   321A 3004            JR      NC,3220H      :Nein: ok
   321C 11E803          LD      DE,03E8H      :Ja: 24 * 40 addieren damit der
   321F 19              ADD     HL,DE         :Cursor unten wieder erscheint
   3220 18BA            JR      31DCH         :Cursor 'ein'
```

```
; RENUM Einsprung von 31B8H
; I: HL = PTZ

  3222 2B              DEC     HL           ;PTZ -1
  3223 D7              RST     10H          ;Zeichen angegeben ?
  3224 2006            JR      NZ,322CH     ;Ja: weiter bei 322CH
  3226 110A00          LD      DE,000AH     ;Nein: Start-ZN = 10
  3229 D5              PUSH    DE           ;DE retten
  322A 1813            JR      323FH        ;weiter bei 323FH

; Startzeilennummer und Abstand angegeben

  322C D24A1E          JP      NC,1E4AH     ;FC-Error wenn keine Ziffer
  322F CD5A1E          CALL    1E5AH        ;DE = Start-ZN
  3232 CF              RST     08H          ;Danach muß ein Komma folgen
  3233 2C              DEFB    ','-
  3234 30F6            JR      NC,322CH     ;FC-Error wenn nach dem Komma
                                            ;keine Ziffer folgt
  3236 D5              PUSH    DE           ;Start-ZN retten
  3237 CD5A1E          CALL    1E5AH        ;DE = Abstand
  323A 7A              LD      A,D          ;Abstand = 0 ?
  323B B3              OR      E
  323C CA4A1E          JP      Z,1E4AH      ;Ja: FC-Error
  323F ED53E440        LD      (40E4H),DE   ;Abstand retten
  3243 D1              POP     DE           ;Start-ZN zurück
  3244 ED53E240        LD      (40E2H),DE   ;und abspeichern
  3248 FD2AF940        LD      IY,(40F9H)   ;IY = Endadresse des Programms
  324C 110001          LD      DE,0100H     ;DE = 256
  324F FD19            ADD     IY,DE        ;IY = IY + 256
  3251 FDE5            PUSH    IY           ;Endadresse + 256 retten
  3253 2AA440          LD      HL,(40A4H)   ;HL = Startadresse des Programms
  3256 E5              PUSH    HL           ;Startadresse retten

; RENUM-Tabelle aufbauen

  3257 7E              LD      A,(HL)       ;Ende des Programms erreicht ?
  3258 23              INC     HL           ;Zeiger +1
  3259 B6              OR      (HL)         ;nächster Zeilenpointer = 0 ?
  325A 284A            JR      Z,32A6H      ;Ja: weiter bei 32A6H
  325C 23              INC     HL           ;Zeiger +2
  325D 23              INC     HL
  325E CDBA33          CALL    33BAH        ;Zeile nach GOTO, GOSUB etc
                                            ;absuchen
  3261 23              INC     HL           ;Zeiger +1
  3262 28F3            JR      Z,3257H      ;nächste Zeile bearbeiten wenn
                                            ;kein Keyword gefunden wurde
  3264 CDD833          CALL    33D8H        ;ZN angegeben ?
                                            ;(Bei THEN und ELSE kann entweder
                                            ;ein Befehl oder eine Zeilen-
                                            ;nummer stehen !)
```

```
3267 2B                 DEC    HL              ;Zeiger -1
3268 28F4               JR     Z,325EH         ;Nein: weitersuchen
326A 23                 INC    HL              ;Zeiger +1
326B E5                 PUSH   HL              ;Programmzeiger retten
326C D5                 PUSH   DE              ;Tabellenzeiger retten
326D FDE5               PUSH   IY              ;Endadresse des Programms retten
326F D1                 POP    DE              ;und zurück nach DE
3270 2AB140             LD     HL,(40B1H)      ;HL = Höchster Speicherplatz
3273 ED52               SBC    HL,DE           ;Tabellenzeiger davon abziehen.
                                               ;Noch Platz im Speicher ?
3275 DA7A19             JP     C,197AH         ;Nein: OM-Error
3278 110400             LD     DE,0004H        ;Ja: Darüberhinaus noch 4 Bytes
327B ED52               SBC    HL,DE           ;frei ?
327D DA7A19             JP     C,197AH         ;Nein: OM-Error
3280 FD7000             LD     (IY+00H),B      ;Ja: Anzahl der Ziffern der
                                               ;gefundenen ZN abspeichern
3283 E1                 POP    HL              ;Tabellenzeiger zurück
3284 CD5A1E             CALL   1E5AH           ;DE = Gefundene ZN
3287 FD7301             LD     (IY+01H),E      ;Zeilennummer abspeichern
328A FD7202             LD     (IY+02H),D
328D FD360300           LD     (IY+03H),00H    ;mit 00H markieren
3291 CDB133             CALL   33B1H           ;IY um 4 erhöhen
3294 E1                 POP    HL              ;Programmzeiger zurück
3295 2B                 DEC    HL              ;Zeiger -1
3296 23                 INC    HL              ;Zeiger +1
3297 7E                 LD     A,(HL)          ;A = nächstes Zeichen
3298 FE20               CP     20H             ;Leerzeichen gefunden ?
329A 28FA               JR     Z,3296H         ;Ja: nächstes Zeichen holen
329C FE2C               CP     2CH             ;Komma gefunden (bei ON GOTO bzw
                                               ;ON GOSUB !)
329E 2803               JR     Z,32A3H         ;Ja: Wurde danach noch eine ZN
                                               ;angegeben ?
32A0 2B                 DEC    HL              ;Zeiger -1
32A1 18BB               JR     325EH           ;Zeile weiter absuchen

; ',' gefunden

32A3 23                 INC    HL              ;Zeiger +1
32A4 18BE               JR     3264H           ;nächste ZN suchen

; RENUM-Tabelle fertig
; Jetzt die neuen Zeilennummern in die Tabelle einsetzen

32A6 FD3600FF           LD     (IY+00H),0FFH;Tabelle abschließen
32AA E1                 POP    HL              ;Programmstart und
32AB FDE1               POP    IY              ;Tabellenstart zurück
32AD ED5BE240           LD     DE,(40E2H)      ;DE = Start-ZN
32B1 D5                 PUSH   DE              ;Neue ZN retten
32B2 FDE5               PUSH   IY              ;Tabellenstart retten
32B4 E5                 PUSH   HL              ;Programmstart retten
32B5 D5                 PUSH   DE              ;Start-ZN retten
32B6 CDC209             CALL   09C2H           ;BCDE = (HL): DE = Zeiger zur
                                               ;nächsten Zeile, BC = Zeilen-
                                               ;nummer
```

```
32B9 7A              LD     A,D            ;Programmende erreicht ?
32BA B3              OR     E
32BB 2841            JR     Z,32FEH        ;Ja: weiter bei 32FEH
32BD EB              EX     DE,HL          ;HL = Zeiger zur nächsten Zeile
32BE D1              POP    DE             ;Neue ZN zurück
32BF FDE5            PUSH   IY             ;Tabellenzeiger retten

32C1 FD7E00          LD     A,(IY+00H)     ;Tabellenende erreicht ?
32C4 3C              INC    A              ;(FFH +1 = 00H)
32C5 2821            JR     Z,32E8H        ;Ja: weiter bei 32E8H
32C7 FD7E03          LD     A,(IY+03H)     ;A = Tabellenmarker
32CA B7              OR     A              ;Ist der Marker schon <> 0
32CB 2016            JR     NZ,32E3H       ;Ja: Diese Eintrag hat schon die
                                           ;neue Zeilennummer
32CD FD7E01          LD     A,(IY+01H)     ;Nein: Ist die jetzige Zeilen-
32D0 B9              CP     C -            ;nummer schon in der Tabelle
32D1 2010            JR     NZ,32E3H       ;enthalten ?
32D3 FD7E02          LD     A,(IY+02H)
32D6 B8              CP     B
32D7 200A            JR     NZ,32E3H       ;Nein: nächsten Eintrag über-
                                           ;prüfen
32D9 FD7301          LD     (IY+01H),E     ;Ja: Neue Zeilennummer in die
32DC FD7202          LD     (IY+02H),D     ;Tabelle einsetzen
32DF FD360301        LD     (IY+03H),01H   ;und den Eintrag markieren
32E3 CDB133          CALL   33B1H          ;Tabellenzeiger um 4 erhöhen
32E6 18D9            JR     32C1H          ;Nächsten Eintrag überprüfen

; Tabellenende erreicht, nächste Zeilennummer eintragen

32E8 FDE1            POP    IY             ;Tabellenstart zurück
32EA E5              PUSH   HL             ;Programmzeiger zurück
32EB 2AE440          LD     HL,(40E4H)     ;HL = Abstand
32EE 19              ADD    HL,DE          ;HL = Neue ZN + Abstand = Nächste
                                           ;neue ZN
32EF DA7A19          JP     C,197AH        ;Nächste ZN > 65535 ?
                                           ;Ja: OM-Error (falscher Fehler-
                                           ;code !!)
32F2 EB              EX     DE,HL          ;DE = Neue ZN
32F3 21F8FF          LD     HL,0FFF8H      ;Neue ZN > 65529 ?
32F6 ED52            SBC    HL,DE
32F8 DA7A19          JP     C,197AH        ;Ja: OM-Error (?!?)
32FB E1              POP    HL             ;Programmzeiger zurück
32FC 18B7            JR     32B5H          ;Nächste Zeilennummer in der
                                           ;Tabelle suchen
```

```
; Neue Zeilennummern ins Programm einsetzen

  32FE D1           POP    DE           :Stack korrigieren
  32FF E1           POP    HL           :Programmstart zurück
  3300 FDE1         POP    IY           :Tabellenstart zurück
  3302 D1           POP    DE           :Start-ZN zurück
  3303 7E           LD     A,(HL)       :Ende des Programms erreicht ?
  3304 23           INC    HL
  3305 B6           OR     (HL)
  3306 CABA31       JP     Z,31BAH      :Ja: RENUM fertig
  3309 23           INC    HL           :Neue Zeilennummer ins Programm
  330A 73           LD     (HL),E       :einsetzen
  330B 23           INC    HL
  330C 72           LD     (HL),D
  330D CDBA33       CALL   33BAH        :GOTO, GOSUB etc suchen
  3310 23           INC    HL -         :Zeiger +1
  3311 2009         JR     NZ,331CH     :Sprung wenn Token gefunden
  3313 E5           PUSH   HL           :Zeiger retten
  3314 2AE440       LD     HL,(40E4H)   :HL = Abstand
  3317 19           ADD    HL,DE        :Nächste ZN ausrechnen
  3318 EB           EX     DE,HL        :DE = Neue ZN
  3319 E1           POP    HL           :Zeiger zurück
  331A 18E7         JR     3303H        :Neue ZN ins Programm setzen

: GOTO, GOSUB etc. gefunden

  331C E5           PUSH   HL           :Zeiger retten
  331D D5           PUSH   DE           :Neue ZN retten
  331E CDD833       CALL   33D8H        :Gefunden ZN in den Zeilenbuffer
                                        :übertragen und die Anzahl der
                                        :Ziffern ermitteln
  3321 D1           POP    DE           :Neue ZN zurück
  3322 E1           POP    HL           :Zeiger zurück
  3323 2B           DEC    HL           :Zeiger -1
  3324 28E7         JR     Z,330DH      :Nächstes Token suchen wenn keine
                                        :ZN gefunden wurde (Nach THEN
                                        :oder ELSE z.B.)
  3326 23           INC    HL           :Zeiger +1
  3327 7E           LD     A,(HL)       :A = nächstes Zeichen
  3328 FE20         CP     20H          :Leerzeichen gefunden ?
  332A 28FA         JR     Z,3326H      :Ja: HL bis zum nächsten
                                        :Zeichen <> ' ' erhöhen
  332C D5           PUSH   DE           :Neue ZN retten
  332D E5           PUSH   HL           :Zeiger retten
  332E FD6E01       LD     L,(IY+01H)   :HL = Neue ZN für diese Stelle
  3331 FD6602       LD     H,(IY+02H)
  3334 CD9433       CALL   3394H        :Neue ZN im Zeilenbuffer ablegen
                                        :(in ASCII-Format !)
  3337 FD4E00       LD     C,(IY+00H)   :C = Anzahl der Ziffern (= Länge)
                                        :der alten ZN
  333A CDB133       CALL   33B1H        :IY zum nächsten Eintrag erhöhen
  333D EB           EX     DE,HL        :DE = Adresse der neuen ZN im
                                        :Zeilenbuffer
  333E E1           POP    HL           :HL = Programmzeiger
                                        :(zur alten ZN)
  333F CDF333       CALL   33F3H        :Alte ZN an dieser Stelle
                                        :ersetzen
```

```
3342 D1               POP     DE            ;Neue ZN zurück
3343 2B               DEC     HL            ;Zeiger -1
3344 23               INC     HL            ;Zeiger +1
3345 7E               LD      A,(HL)        ;Zeiger bis zum nächsten Zeichen
3346 FE20             CP      20H           ;<> Leerzeichen erhöhen
3348 28FA             JR      Z,3344H
334A FE2C             CP      2CH           ;Komma gefunden ?
334C 2803             JR      Z,3351H       ;Ja: Folgt danach noch eine ZN ?
334E 2B               DEC     HL            ;Nein: Zeiger -1
334F 18BC             JR      330DH         ;Nächstes Token suchen

; ',' gefunden (Bei ON GOTO bzw. ON GOSUB)

3351 23               INC     HL            ;Zeiger +1
3352 18C8             JR      331CH         ;Nächste ZN ersetzen

; B Zeichen aus dem Programm löschen
; I: B  = Anzahl der zu löschenden Zeichen
;    HL = Zeiger auf Programmtext
;         (ab (HL) werden B Bytes aus dem Programmtext entfernt)

3354 D5               PUSH    DE            ;Register retten
3355 C5               PUSH    BC
3356 E5               PUSH    HL
3357 E5               PUSH    HL
3358 D1               POP     DE            ;DE = Programmtextzeiger
3359 D5               PUSH    DE            ;DE zweimal retten
335A D5               PUSH    DE
335B 2AF940           LD      HL,(40F9H)    ;HL = Endadresse des Programms
335E E5               PUSH    HL            ;Endadresse retten
335F 2B               DEC     HL            ;Endadresse -1
3360 13               INC     DE            ;Programmzeiger +1
3361 10FC             DJNZ    335FH         ;weiter bis B = 0
3363 22F940           LD      (40F9H),HL    ;Neue Endadresse abspeichern
3366 E1               POP     HL            ;Alte Endadresse zurück
3367 C1               POP     BC            ;Programmzeiger zurück
3368 ED42             SBC     HL,BC         ;Anzahl der Bytes die verschoben
336A 23               INC     HL            ;werden müssen errechnen
336B E5               PUSH    HL            ;Anzahl retten
336C C1               POP     BC            ;und nach BC zurück
336D E1               POP     HL            ;Programmzeiger zurück (2. mal)
336E EB               EX      DE,HL         ;DE = Programmzeiger auf zu
                                            ;löschende Zeichen.
                                            ;HL = Programmzeiger auf rest-
                                            ;lichen Programmtext (nach den zu
                                            ;löschenden Zeichen)
336F EDB0             LDIR                  ;Programmtext verschieben
                                            ;(BC = Zähler)
3371 E1               POP     HL            ;Register zurück
3372 C1               POP     BC
3373 D1               POP     DE
3374 C9               RET
```

```
; B Zeichen in den Programmtext einfügen
; I: B  = Anzahl der einzufügenden Zeichen
;    HL = Zeiger auf Programmtext
;         (ab (HL) können B Bytes in den Programmtext eingefügt werden)

  3375 D5              PUSH   DE             ;Register retten
  3376 C5              PUSH   BC
  3377 E5              PUSH   HL
  3378 2AF940          LD     HL,(40F9H)     ;HL = Endadresse des Programms
  337B E5              PUSH   HL             ;HL retten
  337C D1              POP    DE             ;und zurück nach DE
  337D 23              INC    HL             ;Endadresse +1
  337E 10FD            DJNZ   337DH          ;weiter bis B = 0
  3380 22F940          LD     (40F9H),HL     ;Neue Endadresse retten
  3383 C1              POP    BC             ;Programmtextzeiger zurück
  3384 C5              PUSH   BC -           ;und wieder retten
  3385 E5              PUSH   HL             ;Endadresse retten
  3386 B7              OR     A              ;CY = 0
  3387 ED42            SBC    HL,BC          ;Anzahl der restlichen Zeichen
                                             ;bis zum Programmende errechnen
  3389 E5              PUSH   HL             ;Anzahl retten
  338A C1              POP    BC             ;und zurück nach BC
  338B 03              INC    BC             ;Anzahl +1
  338C E1              POP    HL             ;Neue Endadresse zurück
  338D EB              EX     DE,HL          ;nach DE
  338E EDB8            LDDR                  ;Programm verschieben
  3390 E1              POP    HL             ;Register zurück
  3391 C1              POP    BC
  3392 D1              POP    DE
  3393 C9              RET

; Zeilennummer ins ASCII-Format umwandeln und im Zeilenbuffer ablegen
; I: HL = Zeilennummer
; O: B  = Anzahl der Ziffern (Länge der Zeilennummer)
;    HL = Zeiger auf die Zeilennummer im ASCII-Format

  3394 D5              PUSH   DE             ;DE retten
  3395 222141          LD     (4121H),HL     ;Nummer in X ablegen
  3398 010000          LD     BC,0000H       ;Keine Formatierung
  339B 2AA740          LD     HL,(40A7H)     ;HL = Adresse des Zeilenbuffers
  339E E5              PUSH   HL             ;Adresse retten
  339F CD2F13          CALL   132FH          ;Zahl decodieren
  33A2 E1              POP    HL             ;Adresse zurück
  33A3 0605            LD     B,05H          ;B = maximale Stellenzahl
  33A5 7E              LD     A,(HL)         ;A = nächste Ziffer
  33A6 D630            SUB    30H            ;Führende Null ?
  33A8 2005            JR     NZ,33AFH       ;Nein: Fertig
  33AA 23              INC    HL             ;Ja: Zeiger erhöhen
  33AB 10F8            DJNZ   33A5H          ;Nächste Ziffer
  33AD 2B              DEC    HL             ;Alle Ziffern = '0': Zeiger -1
                                             ;(zeigt auf eine '0')
  33AE 04              INC    B              ;Zähler +1 (1 Ziffer)
  33AF D1              POP    DE             ;DE zurück
  33B0 C9              RET
```

```
; Tabellenzeiger auf den nächsten Eintrag erhöhen

  33B1 FD23              INC     IY
  33B3 FD23              INC     IY
  33B5 FD23              INC     IY
  33B7 FD23              INC     IY
  33B9 C9                RET

; Zeile ab (HL) nach GOTO, GOSUB etc absuchen
; I: HL = Zeiger auf Zeilentext
; O: A  = gefundenes Token
;    Z  = 1 kein Token gefunden
;    HL = Zeiger auf gefundenes Token oder Zeilenende

  33BA 23                INC     HL          ;Zeiger +1
  33BB 7E                LD      A,(HL)      ;A = nächstes Zeichen
  33BC B7                OR      A           ;Zeilenende erreicht ?
  33BD C8                RET     Z           ;Ja: Fertig
  33BE FE8D              CP      8DH         ;'GOTO'-Token ?
  33C0 280C              JR      Z,33CEH     ;Ja: weiter bei 33CEH
  33C2 FE91              CP      91H         ;'GOSUB'-Token ?
  33C4 2808              JR      Z,33CEH     ;Ja: weiter bei 33CEH
  33C6 FECA              CP      0CAH        ;'THEN'-Token ?
  33C8 2804              JR      Z,33CEH     ;Ja: weiter bei 33CEH
  33CA FE95              CP      95H         ;'ELSE'-Token ?
  33CC 20EC              JR      NZ,33BAH    ;Nein: nächstes Zeichen

; Token gefunden

  33CE 2B                DEC     HL          ;War es ein Colour Basic Token ?
  33CF 7E                LD      A,(HL)      ;War das vorherige Zeichen
  33D0 FEFF              CP      0FFH        ;gleich 0FFH ?
  33D2 23                INC     HL          ;Zeiger +1
  33D3 7E                LD      A,(HL)      ;Zeichen wieder zurück
  33D4 28E4              JR      Z,33BAH     ;Ja: weitersuchen
  33D6 A7                AND     A           ;Nein: Z = 0
  33D7 C9                RET

; Länge der gefundenen Zeilennummer feststellen und Zeilennummer
; in den Zeilenbuffer übertragen
; I: HL = Zeiger auf Zeilennummer (auf gefundenes Token + 1)
; O: B  = Anzahl der Ziffern (Länge der Zeilennummer)
;    DE = Zeiger auf Zeilenbuffer (ab (DE) steht die Zeilennummer)
;    HL = Zeiger auf nächstes Zeichen nach der Zeilennummer

  33D8 ED5BA740          LD      DE,(40A7H)  ;DE = Adresse des Zeilenbuffers
  33DC D5                PUSH    DE          ;Adresse retten
  33DD 0600              LD      B,00H       ;Zähler auf 0
  33DF 7E                LD      A,(HL)      ;nächstes Zeichen holen
  33E0 FE20              CP      20H         ;Leerzeichen gefunden ?
  33E2 280B              JR      Z,33EFH     ;Ja: nächstes Zeichen
```

```
 33E4 FE30            CP     30H         :Ziffer gefunden ?
 33E6 380A            JR     C,33F2H     :Nein: Fertig
 33E8 FE3A            CP     3AH         ;Ziffer ?
 33EA 3006            JR     NC,33F2H    :Nein: Fertig
 33EC 04              INC    B           ;Ja: Zähler +1
 33ED 12              LD     (DE),A      ;Ziffer ablegen
 33EE 13              INC    DE          :Bufferzeiger +1
 33EF 23              INC    HL          :Zeilenzeiger +1
 33F0 18ED            JR     33DFH       :Nächstes Zeichen

 33F2 AF              XOR    A           ;A = 00H
 33F3 12              LD     (DE),A      ;Ende der Zeilennummer im Buffer
                                         ;markieren
 33F4 D1              POP    DE          :Bufferadresse zurück
 33F5 04              INC    B           :Z-Flag setzen
 33F6 05              DEC    B           ;Z = 0 wenn Ziffern gefunden
 33F7 C9              RET

; Programmtext so verschieben, daß die neue Zeilennummer eingepasst werden kann
; und die neue Zeilennummer einsetzen
; I: B  = Länge der neuen Zeilennummer
;    C  = Länge der alten Zeilennummer
;    HL = Zeiger auf Programmtext
;         (ab (HL) wird die neue Zeilennummer eingefügt)
;    DE = Zeiger auf die neuen Zeilennummer (im ASCII-Format)

 33F8 C5              PUSH   BC          :Längen retten
 33F9 78              LD     A,B         ;A = neue Länge
 33FA 99              SBC    A,C         ;- alte Länge
 33FB 2810            JR     Z,340DH     ;Zeilennummer einfügen wenn beide
                                         ;Längen identisch sind
 33FD 05              DEC    B           ;Neue Länge -1
 33FE 2808            JR     Z,3408H     ;weiter bei 3408H wenn alte Länge
                                         ;größer ist
 3400 0D              DEC    C           ;Alte Länge -1
 3401 20F6            JR     NZ,33F9H    ;weiter bei 33F9H wenn alte Länge
                                         ;kleiner ist
 3403 CD7533          CALL   3375H       :Zeichen einfügen
 3406 1805            JR     340DH       :und Zeilennummer übertragen

; Alte Länge ist größer als die neue Länge

 3408 41              LD     B,C         :B = Restlänge
 3409 05              DEC    B           ;B -1
 340A CD5433          CALL   3354H       :Zeichen löschen
 340D C1              POP    BC          :Längen zurück
 340E 1A              LD     A,(DE)      :Ziffer vom Zeilenbuffer
 340F 77              LD     (HL),A      :in den Programmtext über-
                                         ;tragen
 3410 13              INC    DE          :Bufferzeiger +1
 3411 23              INC    HL          :Programmzeiger +1
 3412 10FA            DJNZ   340EH       :B Bytes übertragen
 3414 C9              RET
```

```
; Taste holen und FKEY Überprüfen (siehe 3000H)

   3415 CD4900          CALL    0049H       ;Auf Tastendruck warten
   3418 FE5C            CP      5CH         ;FKEY-Taste gedrückt ?
   341A 383C            JR      C,3458H     ;Nein: weiter bei 3458H
   341C FE60            CP      60H         ;FKEY-Taste ?
   341E 3038            JR      NC,3458H    ;Nein: weiter bei 3458H
   3420 D65B            SUB     5BH         ;Ja: A = 01 bis 04 für
                                            ;FKEY1 bis 4

; FKEY-Text in Zeilenbuffer ablegen
; A = Code des gewünschten FKEY-Textes (01 bis 08 für FKEY1 bis FKEY8)

   3422 57              LD      D,A         ;D = Code
   3423 78              LD      A,B         ;A = maximale Zeichenzahl
                                            ;(siehe 05D9H ff)
   3424 FE07            CP      07H         ;noch 7 Zeichen erlaubt ?
   3426 38ED            JR      C,3415H     ;Nein: neue Taste holen
   3428 7A              LD      A,D         ;Code zurück nach A
   3429 C5              PUSH    BC          ;Zähler retten
   342A E5              PUSH    HL          ;Bufferzeiger retten
   342B 214943          LD      HL,4349H    ;HL = Zeiger auf FKEY-Texte - 7
   342E 110700          LD      DE,0007H    ;DE = Länge eines Textes
   3431 19              ADD     HL,DE       ;HL auf nächsten Text erhöhen
   3432 3D              DEC     A           ;Code -1, gewünschten Text
                                            ;erreicht ?
   3433 20FC            JR      NZ,3431H    ;Nein: nächsten Text
   3435 D1              POP     DE          ;Bufferzeiger zurück
   3436 0607            LD      B,07H       ;7 Zeichen übertragen
   3438 7E              LD      A,(HL)      ;Zeichen holen
   3439 FE00            CP      00H         ;Soll 'RETURN' eingefügt werden ?
   343B 2812            JR      Z,344FH     ;Ja: weiter bei 344FH
   343D 12              LD      (DE),A      ;Zeichen ablegen
   343E D5              PUSH    DE          ;Bufferzeiger retten
   343F CD3300          CALL    0033H       ;Zeichen darstellen
   3442 D1              POP     DE          ;Bufferzeiger zurück
   3443 23              INC     HL          ;Textzeiger +1
   3444 13              INC     DE          ;Bufferzeiger +1
   3445 10F1            DJNZ    3438H       ;Nächstes Zeichen
   3447 EB              EX      DE,HL       ;HL = Bufferzeiger
   3448 C1              POP     BC          ;Zähler zurück
   3449 78              LD      A,B         ;A = Zähler
   344A D607            SUB     07H         ;7 Zeichen abziehen
   344C 47              LD      B,A         ;Zähler zurück
   344D 18C6            JR      3415H       ;Nächsten Tastendruck abwarten
```

```
: 'RETURN' einfügen

  344F EB           EX     DE,HL       :HL = Bufferzeiger
  3450 78           LD     A,B         :A = Restlänge des FKEY-Textes
  3451 C1           POP    BC          :Zähler zurück
  3452 80           ADD    A,B         :A = maximale Zeichenzahl
                                       :+ Restlänge
  3453 D607         SUB    07H         :7 Zeichen abziehen
  3455 47           LD     B,A         :neuen Zähler zurück
  3456 3E0D         LD     A,0DH       :A = 'RETURN'-Zeichen

: Shift-FKEY gedrückt ? (FKEY5 bis FKEY8)

  3458 FE7C         CP     7CH         :FKEY gedrückt ?
  345A DA0330       JP     C,3003H     :Nein: weiter bei 3003H
  345D FE80         CP     80H         :FKEY ?
  345F D20330       JP     NC,3003H    :Nein: weiter bei 3003H
  3462 D677         SUB    77H         :A = Code (05H bis 08H)
  3464 18BC         JR     3422H       :weiter bei 3422H

: FKEY

  3466 2B           DEC    HL          :PTZ -1
  3467 D7           RST    10H         :Nächstes Zeichen holen
  3468 D24A1E       JP     NC,1E4AH    :FC-Error wenn keine Ziffer
  346B D630         SUB    30H         :A = Zifferwert (1 bis 8)
  346D F5           PUSH   AF          :Ziffernwert retten
  346E 23           INC    HL          :PTZ +1
  346F CF           RST    08H         :Nächstes Zeichen muß '=' sein
  3470 D5           DEFB   '='-Token
  3471 CF           RST    08H         :Jetzt muß eine Stringkonstante
  3472 22           DEFB   '"'         :folgen
  3473 F1           POP    AF          :Ziffernwert zurück
  3474 FE01         CP     01H         :Ziffernwert < 1 ?
  3476 DA4A1E       JP     C,1E4AH     :Ja: FC-Error
  3479 FE09         CP     09H         :Ziffernwert > 8 ?
  347B 30EB         JR     NC,3468H    :Ja: FC-Error
  347D E5           PUSH   HL          :PTZ retten
  347E 214943       LD     HL,4349H    :HL = Zeiger auf FKEY-Texte - 7
  3481 110700       LD     DE,0007H    :DE = Länge eines FKEY-Textes
  3484 19           ADD    HL,DE       :HL auf gewünschten FKEY-Text
                                       :erhöhen
  3485 3D           DEC    A           :Zähler -1
  3486 20FC         JR     NZ,3484H    :Nächsten Text
  3488 EB           EX     DE,HL       :DE = Zeiger auf FKEY-Text
  3489 E1           POP    HL          :PTZ zurück
  348A 2B           DEC    HL          :PTZ -1
```

```
348B 0607               LD      B,07H        :7 Zeichen übertragen
348D D7                 RST     10H          :Nächstes Zeichen holen
348E FE22               CP      22H          :Ende erreicht ?
3490 280C               JR      Z,349EH      :Ja: weiter bei 349EH
3492 FE00               CP      00H          :'RETURN' einfügen ?
3494 280A               JR      Z,34A0H      :Ja: weiter bei 34A0H
3496 12                 LD      (DE),A       :Nein: Zeichen ablegen
3497 13                 INC     DE           :Zeiger +1
3498 10F3               DJNZ    348DH        :Nächstes Zeichen
349A 23                 INC     HL           :PTZ +1
349B CF                 RST     08H          :Das letzte Zeichen muß '"' sein
349C 22                 DEFB    '"'
349D C9                 RET
```

: Textende erreicht

```
349E 3E20               LD      A,20H        :Restliche Zeichen durch
34A0 12                 LD      (DE),A       :Leerzeichen ersetzen
34A1 05                 DEC     B            :Fertig ?
34A2 2803               JR      Z,34A7H      :Ja: weiter bei 34A7H
34A4 13                 INC     DE           :Nein: Zeiger +1
34A5 18F9               JR      34A0H        :Nächstes Zeichen löschen
34A7 B7                 OR      A            :War das letzte Zeichen 00H ?
34A8 C8                 RET     Z            :Ja: Fertig
34A9 18F0               JR      349BH        :Nein: auf '"' überprüfen
```

: FKEY-Texte (diese Tabelle wird ins RAM ab 4350H kopiert)

```
34AB 4C 49 53 54 20 20 20                    :FKEY1: LIST
34B2 52 55 4E 20 20 20 20                    :FKEY2: RUN
34B9 41 55 54 4F 20 20 20                    :FKEY3: AUTO
34C0 45 44 49 54 20 20 20                    :FKEY4: EDIT
34C7 52 45 4E 55 4D 20 20                    :FKEY5: RENUM
34CE 53 59 53 54 45 4D 00                    :FKEY6: SYSTEM <RETURN>
34D5 43 4C 4F 41 44 20 20                    :FKEY7: CLOAD
34DC 43 53 41 56 45 20 22                    :FKEY8: CSAVE"
```

: &

```
34E3 D7                 RST     10H          :nächstes Zeichen holen
34E4 FE48               CP      48H          :Ist es 'H' ?
34E6 203B               JR      NZ,3523H     :Nein: Dann muß es 'O' sein
```

: Hexadezimale Konstante decodieren

```
34E8 23                 INC     HL           :PTZ +1
34E9 CDF334             CALL    34F3H        :DE = Konstante
34EC E5                 PUSH    HL           :PTZ retten
34ED EB                 EX      DE,HL        :HL = Konstante
34EE CD9A0A             CALL    0A9AH        :Als INT nach X laden
34F1 E1                 POP     HL           :PTZ zurück
34F2 C9                 RET
```

```
; DE = &H (HL)

  34F3 110000           LD      DE,0000H      :Ergebnis = 0
  34F6 CD0135           CALL    3501H         :MSB decodieren
  34F9 57               LD      D,A           :D = MSB
  34FA 23               INC     HL            :PTZ +1
  34FB CD0135           CALL    3501H         :LSB decodieren
  34FE 5F               LD      E,A           :E = LSB
  34FF 23               INC     HL            :PTZ +1
  3500 C9               RET

; A = &H (HL)

  3501 CD0F35           CALL    350FH         :Half-Byte decodieren
  3504 07               RLCA                  :In Bits 4-7 schieben
  3505 07               RLCA
  3506 07               RLCA
  3507 07               RLCA
  3508 23               INC     HL            :PTZ +1
  3509 47               LD      B,A           :In B zwischenspeichern
  350A CD0F35           CALL    350FH         :2. Half-Byte decodieren
  350D 80               ADD     A,B           :und letztes Zwischenergebnis
                                              :addieren
  350E C9               RET

; Eine Hexadezimalziffer decodieren

  350F 7E               LD      A,(HL)        :A = Zeichen
  3510 D630             SUB     30H           :Zeichen < '0' ?
  3512 DA4A1E           JP      C,1E4AH       :Ja: FC-Error
  3515 FE0A             CP      0AH           :Zeichen > '9' ?
  3517 D8               RET     C             :Nein: Wert ok
  3518 D607             SUB     07H           :Ja: Zeichen < 'A' ?
  351A DA4A1E           JP      C,1E4AH       :Ja: FC-Error
  351D FE10             CP      10H           :Zeichen > 'F' ?
  351F D24A1E           JP      NC,1E4AH      :Ja: FC-Error
  3522 C9               RET

; &O

  3523 FE4F             CP      4FH           :'O' gefunden ?
  3525 C29719           JP      NZ,1997H      :Nein: SN-Error

; Oktale Konstante decodieren

  3528 110000           LD      DE,0000H      :Ergebnis = 0
  352B 23               INC     HL            :PTZ +1
  352C CD4535           CALL    3545H         :Höchste Ziffer decodieren
  352F 07               RLCA                  :Bits 0 - 2 freischieben
  3530 07               RLCA
  3531 07               RLCA
  3532 23               INC     HL            :PTZ +1
  3533 47               LD      B,A           :Zwischenergebnis nach B
```

```
3534 CD4F35            CALL    354FH        :nächste Ziffer decodieren
3537 80                ADD     A,B          :zum Zwischenergebnis addieren
3538 07                RLCA                 :Bits 0 - 2 freischieben
3539 07                RLCA
353A 07                RLCA
353B 23                INC     HL           :PTZ +1
353C 47                LD      B,A          :Zwischenergebnis nach B
353D CD4F35            CALL    354FH        :letzte Ziffer decodieren
3540 80                ADD     A,B          :zum Zwischenergebnis addieren
3541 5F                LD      E,A          :LSB-Wert nach E
3542 23                INC     HL           :PTZ +1
3543 18A7              JR      34ECH        :Als INT nach X laden

; Höchste Oktalziffer ('0' - '3') decodieren

3545 7E                LD      A,(HL)       :A = Zeichen
3546 D630              SUB     30H          :Zeichen < '0' ?
3548 380D              JR      C,3557H      :Ja: FC-Error
354A FE04              CP      04H          :Zeichen < '4' ?
354C D8                RET     C            :Ja: ok
354D 1808              JR      3557H        :Nein: FC-Error

; Niedere Oktalziffer ('0' - '7') decodieren

354F 7E                LD      A,(HL)       :A = Zeichen
3550 D630              SUB     30H          :Zeichen < '0' ?
3552 3803              JR      C,3557H      :Ja: FC-Error
3554 FE08              CP      08H          :Zeichen < '8' ?
3556 D8                RET     C            :Ja: ok
3557 C34A1E            JP      1E4AH        :Nein: FC-Error

; CALL

355A CDF334            CALL    34F3H        :Hex-Konstante nach DE
355D E5                PUSH    HL           :PTZ retten
355E EB                EX      DE,HL        :HL = Hex-Konstante (Adresse)
355F 116735            LD      DE,3567H     :RET-Adr auf 3567H setzen
3562 D5                PUSH    DE
3563 E9                JP      (HL)         :Routine anspringen

3564 FF                RST     38H          :--
3565 FF                RST     38H
3566 FF                RST     38H

; RET von CALL-Routine

3567 E1                POP     HL           :PTZ zurück
3568 C9                RET
```

```
; Farbcode-Anpassungs-Tabelle (wird ins RAM ab 4390H kopiert)
; Bei den ersten Colour-Genies wurde der COLOUR-Wert direkt ins Farb-RAM
; (ab F000H) geschrieben. Durch Schaltungsänderungen in den neueren Geräten
; wurden jedoch die Farbcodes vertauscht, so daß durch diese Tabelle eine
; Anpassung auf die alten Geräte erreicht werden musste.
; Diese Tabelle gibt nun für jeden COLOUR-Wert den Wert an, der ins Farb-RAM
; geschrieben werden muß um die alte Reihenfolge wieder zu erhalten
;
;      Wert im Farb-RAM        COLOUR-Wert

  3569 03                        1
  356A 05                        2
  356B 02                        3
  356C 04                        4
  356D 06                        5
  356E 08                        6
  356F 01                        7
  3570 0E                        8
  3571 09                        9
  3572 10                       10
  3573 07                       11
  3574 0B                       12
  3575 0C                       13
  3576 0D                       14
  3577 0A                       15
  3578 0F                       16

; Textkonstante ab (HL) und Ton ausgeben (Bei Fehlermeldungen)

  3579 CDA728              CALL    28A7H         ;Text ausgeben
  357C 110900              LD      DE,0009H      ;PSG-Programmierungswerte
  357F D5                  PUSH    DE            ;ins Stack schreiben
  3580 110008              LD      DE,0800H
  3583 D5                  PUSH    DE
  3584 113E10              LD      DE,103EH
  3587 D5                  PUSH    DE
  3588 117800              LD      DE,0078H      ;D und E = PSG-Werte
  358B 3E01                LD      A,01H         ;A = Register (0 und 1)
  358D CD2A3E              CALL    3E2AH         ;SOUND A,D: SOUND A-1,E
  3590 D1                  POP     DE            ;Wert zurück
  3591 3E08                LD      A,08H         ;Register 7 und 8
  3593 CD2A3E              CALL    3E2AH
  3596 D1                  POP     DE            ;Wert zurück
  3597 3E0C                LD      A,0CH         ;Register 11 und 12
  3599 CD2A3E              CALL    3E2AH
  359C D1                  POP     DE            ;Wert zurück
  359D 3E0D                LD      A,0DH         ;Register 13
  359F C3323E              JP      3E32H         ;SOUND A,E

; UPRO für SHAPE (siehe 3CF2H ff)
; Startadresse der SHAPE-Tabelle nach HL laden

  35A2 2AB140              LD      HL,(40B1H)    ;HL = Start der SHAPE-Tabelle
  35A5 23                  INC     HL            ;+1
  35A6 C3F53C              JP      3CF5H         ;weiter bei SHAPE
```

```
: UPRO für PRINT#

  35A9 CDB535         CALL    35B5H       ;Zahl decodieren
  35AC C33F02         JP      023FH       ;Leader und Sync schreiben

: UPRO für INPUT#

  35AF CDB535         CALL    35B5H       ;Zahl decodieren
  35B2 C34C02         JP      024CH       ;Leader und Sync suchen

: Zahl bei PRINT# und INPUT# decodieren

  35B5 AF             XOR     A           ;A = 0
  35B6 CD012B         CALL    2B01H       ;Zahl nach DE holen
  35B9 CF             RST     08H         ;Danach muß ein Komma folgen
  35BA 2C             DEFB    ',' -
  35BB 7B             LD      A,E         ;A = LSB
  35BC A2             AND     D           ;mit MSB verknüpfen (MSB muß FFH
                                          ;sein, da eine negative Zahl
                                          ;verlangt wird)
  35BD C602           ADD     A,02H       ;War die Zahl -1 oder -2 ?
  35BF D24A1E         JP      NC,1E4AH    ;Nein: FC-Error
  35C2 C9             RET                 ;Ja: ok

: Lautsärke des angegebenen Kanals auf 0 setzen (unbenutzt)
: I: HL = PTZ auf Argument oder Befehls- bzw. Zeilenende
: Liegt Wert des Argument zwischen 1 und 3, dann wird die Lautstärke des
: entsprechenden Kanal des PSG auf Null gesetzt.
: Ist kein Argument angegeben, so wird die Lautstärke aller Kanäle auf Null
: gesetzt
: Diese Routine kann mit Hilfe der folgenden Befehle über den KILL-Befehl
: angesprochen werden. (also durch KILL n oder KILL)
: POKE &H4192 , 195 : POKE &H4193 , 53

  35C3 2B             DEC     HL          ;PTZ -1
  35C4 D7             RST     10H         ;Kanalnummer angegeben ?
  35C5 281A           JR      Z,35E1H     ;Nein: weiter bei 35E1H
  35C7 CD1C2B         CALL    2B1CH       ;Ja: Zahl nach A
  35CA 0608           LD      B,08H       ;Register = 8
  35CC FE01           CP      01H         ;Zahl = 1 ?
  35CE 280B           JR      Z,35DBH     ;Ja: Registernummer ok
  35D0 04             INC     B           ;Register = 9
  35D1 FE02           CP      02H         ;Zahl = 2 ?
  35D3 2806           JR      Z,35DBH     ;Ja: Registernummer ok
  35D5 04             INC     B           ;Register = 10
  35D6 FE03           CP      03H         ;Zahl = 3 ?
  35D8 C24A1E         JP      NZ,1E4AH    ;Nein: FC-Error
  35DB 78             LD      A,B         ;A = Registernummer
  35DC 1E00           LD      E,00H       ;E = 00
  35DE C3323E         JP      3E32H       ;SOUND A,E (Lautstärke des ange-
                                          ;gebenen Kanals auf 0 setzen)
```

```
; Kein Kanal angegeben: Lautstärke aller Kanäle auf 0 setzen

  35E1 3E0A          LD     A,0AH        ;Lautstärke der Kanäle 2 und 3
  35E3 110000        LD     DE,0000H     ;auf 0 setzen
  35E6 CD2A3E        CALL   3E2AH
  35E9 3E08          LD     A,08H        ;Lautstärke des Kanals 1 auch
  35EB C3323E        JP     3E32H        ;auf 0 setzen

; UPRO für die Bildschirmausgabe
; Zeichen in A bei (HL) darstellen und Zeichenfarbe setzen

  35EE 77            LD     (HL),A       ;Zeichen darstellen
  35EF CD7A30        CALL   307AH        ;Zeichenfarbe setzen
  35F2 23            INC    HL           ;Adresse +1 für Cursor
  35F3 CD7A30        CALL   307AH        ;Cursorfarbe setzen
  35F6 2B            DEC    HL -         ;Adresse -1
  35F7 C9            RET

; UPRO für die Bildschirmausgabe
; Neuen Farbcode ins Farb-RAM setzen

  35F8 1100AC        LD     DE,0AC00H    ;DE = Offset zum Farb-RAM
                                         ;(Bildschirmadresse + DE
                                         ;= zugehörige Farb-RAM-Adresse)
  35FB FE20          CP     20H          ;Zeichen = Leerzeichen ?
  35FD C28030        JP     NZ,3080H     ;Nein: Neue Farbe setzen
  3600 3A9043        LD     A,(4390H)    ;Ja: A = COLOURO-Code
  3603 19            ADD    HL,DE        ;HL = Farb-RAM-Adresse
  3604 C38F30        JP     308FH        ;Bei Leerzeichen COLOURO setzen

; UPRO für 'Cursor eine Zeile tiefer'

  3607 112800        LD     DE,0028H     ;DE = 40 (Zeilenlänge)
  360A 3E20          LD     A,20H        ;A = Leerzeichen
  360C C37A30        JP     307AH        ;Leerzeichen ausgeben
                                         ;(Cursorzeichen löschen)

; UPRO für 'Cursor eine Zeile höher'

  360F 11D8FF        LD     DE,0FFD8H    ;DE = -40 (Zeilenlänge)
  3612 18F6          JR     360AH        ;Leerzeichen ausgeben

; UPRO für 'Cursor ein/aus'

  3614 2002          JR     NZ,3618H     ;Sprung nach 3618H (siehe 306EH)

  3616 3E20          LD     A,20H        ;--
  3618 C37A30        JP     307AH        ;Zeichen in A ausgeben

; UPRO für 'Carriage Return'

  361B CD0A36        CALL   360AH        ;Cursorzeichen löschen
  361E C36531        JP     3165H        ;weiter bei 3165H
```

```
: UPRO für COLOUR-Änderung (siehe 3017H)

  3621 322340            LD      (4023H).A      :Neuen COLOUR-Wert abspeichern
  3624 2A2040            LD      HL.(4020H)     :HL = Aktuelle Cursoradresse
  3627 C35F30            JP      305FH          :Cursor einschalten

: UPRO für PRINT§ (siehe 2086H)
: Neue Bildschirm-POS errechnen

  362A E5                PUSH    HL             :HL retten
  362B EB                EX      DE.HL          :HL = §-Argument
  362C 112800            LD      DE.0028H       :DE = 40 (Zeilenlänge)
  362F 19                ADD     HL.DE          :Zeilenlänge addieren
  3630 B7                OR      A              :CY = 0
  3631 ED52              SBC     HL.DE          :Zeilenlänge solange abziehen.
  3633 30FC              JR      NC.3631H       :bis ein Unterlauf ergfolgt
  3635 19                ADD     HL.DE          :Zeilenlänge addieren
  3636 7D                LD      A.L            :L = Aktuelle POS
  3637 E1                POP     HL             :HL zurück
  3638 C9                RET

: UPRO für Scroll (siehe 3196H ff)
: Register vorbereiten

  3639 D9                EXX                    :Registersatz tauschen
  363A F1                POP     AF             :RET-Adr nach AF
  363B C5                PUSH    BC             :Register retten
  363C D5                PUSH    DE
  363D E5                PUSH    HL
  363E 1100F0            LD      DE.0F000H      :2. Registersatz auf
  3641 2128F0            LD      HL.0F028H      :Farbspeicher-Scroll vorbereiten
  3644 01C003            LD      BC.03C0H
  3647 C5                PUSH    BC             :BC als Zähler retten
  3648 D9                EXX                    :Register tauschen
  3649 110044            LD      DE.4400H       :1. Registersatz auf Bildschirm-
  364C 212844            LD      HL.4428H       :speicher-Scroll vorbereiten
  364F C1                POP     BC             :Zähler zurück
  3650 F5                PUSH    AF             :RET-Adr ins Stack
  3651 C9                RET

  JOY als Befehl

  3652 7E                LD      A.(HL)         :Nächstes Zeichen holen
  3653 23                INC     HL             :PTZ +1
  3654 FEDB              CP      0DBH           :'INP'-Token gefunden ?
  3656 CAC536            JP      Z.36C5H        :Ja: weiter bei 36C5H
  3659 FEA0              CP      0A0H           :'OUT'-Token gefunden ?
  365B CAB336            JP      Z.36B3H        :Ja: weiter bei 36B3H
  365E C39713            JP      1397H          :Nein: SN-Error
```

```
; COLOUR rechts vom Gleichheitszeichen
; (NIE benutzen !!)

  3661 3A2340        LD     A,(4023H)    :A = COLOUR-Wert
  3664 3C            INC    A            :A = FFH ? (+ 1 = 0 ?)
  3665 CA0D37        JP     Z,370DH      :Ja: 0 ist Ergebnis
  3668 B8            CP     B            :wozu ?
  3669 77            LD     (HL),A       :Wert bei (PTZ) abspeichern
                                         :-> evtl. Programmzerstörung !!

; Basicprogramm vom I/O Port übernehmen (JOYINP)

  366A 2AA440        LD     HL,(40A4H)   :HL = Startadresse des Basic-
  366D 2B            DEC    HL           :programms - 1
  366E 0603          LD     B,03H        :Zähler = 3
  3670 E5            PUSH   HL           :Adresse retten
  3671 260F          LD     H,0FH        :Port B anwählen
  3673 CDBB3A        CALL   3ABBH        :und Wert holen
  3676 CB57          BIT    02H,A        :A.2 = 0  ?
  3678 28F9          JR     Z,3673H      :Ja: Warten bis A.2 = 1
  367A CDBB3A        CALL   3ABBH        :nächsten Wert holen
  367D CB57          BIT    02H,A        :A.2 = 1 ?
  367F 20F9          JR     NZ,367AH     :Ja: Warten bis A.2 wieder 0
  3681 260E          LD     H,0EH        :Port A anwählen
  3683 CDBB3A        CALL   3ABBH        :und Wert holen
  3686 E1            POP    HL           :Adresse zurück
  3687 77            LD     (HL),A       :und Wert abspeichern
  3688 B7            OR     A            :Wert = 0 ?
  3689 23            INC    HL           :Adresse +1
  368A 20E2          JR     NZ,366EH     :Nein: Zähler auf 3 und nächstes
                                         :Zeichen holen
  368C 10E2          DJNZ   3670H        :Ja: 3 mal 0 hintereinander ?
                                         :Nein: nächstes Zeichen holen
  368E C9            RET                 :Ja: Programmende erreicht

; Basicprogramm zum I/O-Port ausgeben (JOYOUT)

  368F 2AA440        LD     HL,(40A4H)   :HL = Startadresse des Basic-
  3692 2B            DEC    HL           :programms - 1
  3693 ED5BF940      LD     DE,(40F9H)   :DE = Endadresse des Basic-
                                         :programms
  3697 7E            LD     A,(HL)       :A = nächstes Zeichen
  3698 E5            PUSH   HL           :Adresse retten
  3699 6F            LD     L,A          :L = Zeichen
  369A 260E          LD     H,0EH        :Port A anwählen
  369C CDB23A        CALL   3AB2H        :und Zeichen ausgeben
  369F 24            INC    H            :Port B anwählen
  36A0 3E04          LD     A,04H        :A.2 = 1 setzen
  36A2 CDB23A        CALL   3AB2H        :A ausgeben
  36A5 060A          LD     B,0AH        :Verzögerungsschleife
  36A7 10FE          DJNZ   36A7H
```

```
36A9 AF            XOR    A            :A,2 = 0 setzen
36AA CDB23A        CALL   3AB2H        :A ausgeben
36AD E1            POP    HL           :Adresse zurück
36AE 23            INC    HL           :Adresse +1
36AF DF            RST    18H          :Programmende erreicht ?
36B0 20E5          JR     NZ.3697H     :Nein: nächstes Zeichen
36B2 C9            RET                 :Ja: Fertig

; JOYOUT

36B3 E5            PUSH   HL           :PTZ retten
36B4 2607          LD     H.07H        :Wert vom PSG-Register 7
36B6 CDBB3A        CALL   3ABBH        :holen
36B9 E63F          AND    3FH          :Bits 6 und 7 ausblenden
36BB F6C0          OR     0C0H         :und beide setzen
36BD CDB23A        CALL   3AB2H        :Port A und B zur Ausgabe
                                       :vorbereiten
36C0 CD8F36        CALL   368FH        :Basicprogramm ausgeben
36C3 E1            POP    HL           :PTZ zurück
36C4 C9            RET

; JOYINP

36C5 3E07          LD     A.07H        :Register 7 des PSGs
36C7 D3F8          OUT    (0F8H).A     :anwählen
36C9 3E3F          LD     A.3FH        :und Port A und B zur Eingabe
36CB D3F9          OUT    (0F9H).A     :vorbereiten
36CD CD6A36        CALL   366AH        :Basicprogramm übernehmen
36D0 C3772C        JP     2C77H        :und alle Zeiger neu setzen

; SWAP

36D3 CD0D26        CALL   260DH        :Adresse der 1. Variablen
                                       :ermitteln
36D6 D5            PUSH   DE           :und retten
36D7 3AAF40        LD     A.(40AFH)    :VT der 1. Variablen
36DA F5            PUSH   AF           :retten
36DB CF            RST    08H          :Jetzt muß ein Komma folgen
36DC 2C            DEFB   ','
36DD CD0D26        CALL   260DH        :Adresse der 2. Variablen
                                       :ermitteln
36E0 C1            POP    BC           :VT der 1. Variablen zurück
36E1 3AAF40        LD     A.(40AFH)    :VT der 2. Variablen holen
36E4 B8            CP     B            :Beide gleichen Typs ?
36E5 C24A1E        JP     NZ.1E4AH     :Nein: FC-Error
36E8 E3            EX     (SP).HL      :Ja: PTZ retten. Adresse der
                                       :1. Variablen zurück
36E9 4E            LD     C.(HL)       :Wert von der ersten
36EA 1A            LD     A.(DE)       :und zweiten Variablen holen
36EB 77            LD     (HL).A       :und vertauscht wieder
36EC 79            LD     A.C          :abspeichern
36ED 12            LD     (DE).A
36EE 23            INC    HL           :Beide Zeiger erhöhen
36EF 13            INC    DE
36F0 10F7          DJNZ   36E9H        :VT Bytes übertragen
36F2 E1            POP    HL           :PTZ zurück
36F3 C9            RET
```

```
; SOUND (n)

  36F4 CF            RST    08H          :Klammer auf ?
  36F5 28            DEFB   '('
  36F6 CD1C2B        CALL   2B1CH        ;Wert nach A holen
  36F9 F5            PUSH   AF           ;Wert retten
  36FA CF            RST    08H          :Klammer zu ?
  36FB 29            DEFB   ')'
  36FC F1            POP    AF           :Wert zurück
  36FD FE10          CP     10H          :Wert > 15 ?
  36FF D24A1E        JP     NC,1E4AH     ;Ja: FC-Error
  3702 E5            PUSH   HL           ;PTZ retten
  3703 67            LD     H,A          ;H = Wert
  3704 CDBB3A        CALL   3ABBH        ;Wert des gewünschten
                                         ;PSG-Registers holen
  3707 C3723F        JP     3F72H        ;und als INT in X abspeichern

; SCALE als Funktion

  370A 3A1443        LD     A,(4314H)    ;A = aktuelle Scale-Wert
  370D E5            PUSH   HL           ;PTZ retten
  370E C3723F        JP     3F72H        ;als INT in X abspeichern

; Unbenutzte Farcode-Anpassungstabellen

  3711 10
  3712 0D
  3713 0E
  3714 04
  3715 06
  3716 03
  3717 01
  3718 02
  3719 05
  371A 07
  371B 08
  371C 09
  371D 0A
  371E 0B
  371F 0C
  3720 0F
```

```
3721 10
3722 0D
3723 06
3724 04
3725 0F
3726 03
3727 09
3728 02
3729 01
372A 05
372B 07
372C 08
372D 0A
372E 0B
372F 0C
3730 0E

3731 10
3732 05
3733 02
3734 04
3735 0E
3736 09
3737 01
3738 0A
3739 07
373A 06
373B 0D
373C 03
373D 08
373E 0B
373F 0C
3740 0F

: Unbenutzter ROM-Bereich

3741 FF                 RST     38H
3742 FF                 RST     38H
3743 FF                 RST     38H
3744 FF                 RST     38H
3745 FF                 RST     38H
3746 FF                 RST     38H
3747 FF                 RST     38H
3748 FF                 RST     38H
3749 FF                 RST     38H
374A FF                 RST     38H
374B FF                 RST     38H
374C FF                 RST     38H
374D FF                 RST     38H
```

```
374E FF          RST     38H
374F FF          RST     38H
3750 FF          RST     38H
3751 FF          RST     38H
3752 FF          RST     38H
3753 FF          RST     38H
3754 FF          RST     38H
3755 FF          RST     38H
3756 FF          RST     38H
3757 FF          RST     38H
3758 FF          RST     38H
3759 FF          RST     38H
375A FF          RST     38H
375B FF          RST     38H
375C FF          RST     38H
375D FF          RST     38H
375E FF          RST     38H
375F FF          RST     38H
3760 FF          RST     38H
3761 FF          RST     38H
3762 FF          RST     38H
3763 FF          RST     38H
3764 FF          RST     38H
3765 FF          RST     38H
3766 FF          RST     38H
3767 FF          RST     38H
3768 FF          RST     38H
3769 FF          RST     38H
376A FF          RST     38H
376B FF          RST     38H
376C FF          RST     38H
376D FF          RST     38H
376E FF          RST     38H
376F FF          RST     38H
3770 FF          RST     38H
3771 FF          RST     38H
3772 FF          RST     38H
3773 FF          RST     38H
3774 FF          RST     38H
3775 FF          RST     38H
3776 FF          RST     38H
3777 FF          RST     38H
3778 FF          RST     38H
3779 FF          RST     38H
377A FF          RST     38H
377B FF          RST     38H
377C FF          RST     38H
377D FF          RST     38H
377E FF          RST     38H
377F FF          RST     38H
```

```
3780 FF        RST    38H
3781 FF        RST    38H
3782 FF        RST    38H
3783 FF        RST    38H
3784 FF        RST    38H
3785 FF        RST    38H
3786 FF        RST    38H
3787 FF        RST    38H
3788 FF        RST    38H
3789 FF        RST    38H
378A FF        RST    38H
378B FF        RST    38H
378C FF        RST    38H
378D FF        RST    38H
378E FF        RST    38H
378F FF        RST    38H
3790 FF        RST    38H
3791 FF        RST    38H
3792 FF        RST    38H
3793 FF        RST    38H
3794 FF        RST    38H
3795 FF        RST    38H
3796 FF        RST    38H
3797 FF        RST    38H
3798 FF        RST    38H
3799 FF        RST    38H
379A FF        RST    38H
379B FF        RST    38H
379C FF        RST    38H
379D FF        RST    38H
379E FF        RST    38H
379F FF        RST    38H
37A0 FF        RST    38H
37A1 FF        RST    38H
37A2 FF        RST    38H
37A3 FF        RST    38H
37A4 FF        RST    38H
37A5 FF        RST    38H
37A6 FF        RST    38H
37A7 FF        RST    38H
37A8 FF        RST    38H
37A9 FF        RST    38H
37AA FF        RST    38H
37AB FF        RST    38H
37AC FF        RST    38H
37AD FF        RST    38H
37AE FF        RST    38H
37AF FF        RST    38H
37B0 FF        RST    38H
```

```
37B1 FF                  RST     38H
37B2 FF                  RST     38H
37B3 FF                  RST     38H
37B4 FF                  RST     38H
37B5 FF                  RST     38H
37B6 FF                  RST     38H
37B7 FF                  RST     38H
37B8 FF                  RST     38H
37B9 FF                  RST     38H
37BA FF                  RST     38H
37BB FF                  RST     38H
37BC FF                  RST     38H
37BD FF                  RST     38H
37BE FF                  RST     38H
37BF FF                  RST     38H
37C0 FF                  RST     38H
37C1 FF                  RST     38H
37C2 FF                  RST     38H
37C3 FF                  RST     38H
37C4 FF                  RST     38H
37C5 FF                  RST     38H
37C6 FF                  RST     38H
37C7 FF                  RST     38H
37C8 FF                  RST     38H
37C9 FF                  RST     38H
37CA FF                  RST     38H

; Ausgabe eines Textes auf den Bildschirm (unbenutzt)
; I: HL = Startadresse eines Textes

37CB D5                  PUSH    DE          ;DE retten
37CC 111D40              LD      DE,401DH    ;DE = Bildschirm DCB
37CF 1804                JR      37D5H       ;weiter bei 37D5H

; Ausgabe eines Textes auf den Drucker (unbenutzt)
; Das Ende des Textes muß durch 03H bzw. 0DH markiert werden.
; I: HL = Startadresse eines Textes

37D1 D5                  PUSH    DE          ;DE retten
37D2 112540              LD      DE,4025H    ;DE = Drucker DCB
37D5 E5                  PUSH    HL          ;HL retten
37D6 7E                  LD      A,(HL)      ;Zeichen holen
37D7 FE03                CP      03H         ;Ist es 03H ?
37D9 2809                JR      Z,37E4H     ;Ja: Fertig
37DB CD1B00              CALL    001BH       ;Nein: Zeichen ausgeben
37DE 7E                  LD      A,(HL)      ;War das ausgegebene Zeichen
37DF FE0D                CP      0DH         ;gleich 0DH ?
37E1 23                  INC     HL          ;Zeiger +1
37E2 20F2                JR      NZ,37D6H    ;Nein: Nächstes Zeichen ausgeben
37E4 E1                  POP     HL          ;Ja: HL zurück
37E5 D1                  POP     DE          ;DE zurück
37E6 C9                  RET
```

```
; Umwandlung des Binärwertes in DE in vier HEX-Ziffern (unbenutzt)
; (Umkehrung des &H-Befehls)
; I: DE = Auszugebender Binärwert
;    HL = Zeiger auf Speicherplatz in dem die vier HEX-Ziffern
;         abgelegt werden sollen

  37E7 7A             LD     A,D        ;A = MSB
  37E8 CDEC37         CALL   37ECH      ;MSB umwandeln
  37EB 7B             LD     A,E        ;A = LSB

  37EC F5             PUSH   AF         ;Bits 0-3 retten
  37ED 0F             RRCA              ;Obere vier Bits nach unten
  37EE 0F             RRCA              ;(Bits 0-3) schieben
  37EF 0F             RRCA
  37F0 0F             RRCA
  37F1 CDF537         CALL   37F5H      ;HEX-Ziffer für Bits 0-3 ausgeben
  37F4 F1             POP    AF         ;Bits 0-3 zurück
  37F5 E60F           AND    0FH        ;Bits 0-3 maskieren
  37F7 C690           ADD    A,90H      ;Wert der Bits 0-3 (00H - 0FH)
  37F9 27             DAA               ;in ASCII-Zeichen (HEX-Ziffer)
  37FA CE40           ADC    A,40H      ;('0' - '9' und 'A' - 'F')
  37FC 27             DAA               ;umwandeln
  37FD 77             LD     (HL),A     ;Zeichen ablegen
  37FE 23             INC    HL         ;Zeiger +1
  37FF C9             RET               ;Nächstes Zeichen

; Programmierungstabellen für den CRTC
; Der erste Tabellenwert wird ins Register 15, der letzte ins Register 0
; des CRTCs geschrieben
; Diese Tabellen werden im RAM ab 42F0H abgelegt
; (siehe Anhang A im Handbuch)

; LGR-Modus, PAL-Fernsehnorm

  3800 01
  3801 00
  3802 00
  3803 04
  3804 07
  3805 C4
  3806 07
  3807 A0
  3808 1F
  3809 19
  380A 00
  380B 26
  380C 96
  380D 34
  380E 28
  380F 46
```

```
; FGR-Modus, PAL-Fernsehnorm

  3810 00
  3811 00
  3812 00
  3813 08
  3814 00
  3815 20
  3816 01
  3817 20
  3818 74
  3819 66
  381A 1F
  381B 7E
  381C 96
  381D 34
  381E 28
  381F 46

; Die nächsten drei Werte bestimmen die Geschwindigkeit mit der Zeichen auf
; Cassette geschrieben bzw. davon gelesen werden. Diese drei Werte werden in
; den Speicherstellen 4310H, 4311H und 4312H abgelegt.
; (siehe 01FAH ff und 021FH ff)

  3820 46
  3821 4B
  3822 69

; CRTC Programmierungstabelle für die amerikanische NTSC-Farbnorm.
; Damit das Colour-Genie ein NTSC-Farbsignal ausgibt müssen auch diverse
; Schaltungsänderungen vorgenommen werden. Eine Aufführung dieser Tabelle
; ist also für die europäischen Benutzer wertlos.

; LGR-Modus, NTSC-Farbnorm

  3823 01
  3824 00
  3825 00
  3826 04
  3827 07
  3828 C4
  3829 07
  382A A0
  382B 1B
  382C 19
  382D 06
  382E 1F
  382F 34
  3830 2E
  3831 29
  3832 38
```

```
; FGR-Modus, NTSC-Farbnorm

  3833 00
  3834 00
  3835 00
  3836 08
  3837 00
  3838 20
  3839 01
  383A 20
  383B 6E
  383C 66
  383D 08
  383E 7F
  383F 34
  3840 2E
  3841 28
  3842 38

; Die nächsten drei Bytes bestimmen die Geschwindigkeit von Cassetten-
; operationen bei der amerikanischen Ausgabe des Colour-Genies

  3843 4C
  3844 51
  3845 71

; FCLS ausführen (ohne Parameter)

  3846 3E00          LD     A,00H       ;A = Farbcode (0)
  3848 1808          JR     3852H       ;FCLS ausführen

; Fortsetzung der FCLS-Routine von 3C87H (mit Parameter)

  384A C4C23F        CALL   NZ,3FC2H    ;Wert holen, falls angegeben
  384D FE04          CP     04H         ;Wert > 4 ? (A = Wert - 1)
  384F D24A1E        JP     NC,1E4AH    ;Ja: FC-Error

; UPRO für FCLS (AF,BC,DE,HL)
; FCLS A ausführen
; I:  A   = Farbcode

  3852 4F            LD     C,A         ;Farbwert nach C
  3853 0603          LD     B,03H       ;Die oberen 3 Bit-Paare
  3855 07            RLCA               ;von A auf den angegebenen
  3856 07            RLCA               ;Farbwert setzen
  3857 B1            OR     C
  3858 10FB          DJNZ   3855H       ;Nächstes Bit-Paar
  385A 4F            LD     C,A         ;Wert nach C retten
  385B E5            PUSH   HL          ;PTZ retten
```

```
385C 2AA440         LD     HL,(40A4H)    :HL = Startadresse des Basic-
                                         ;programms
385F 110148         LD     DE,4801H      :DE = Startadresse des Speichers
                                         ;für hochauflösende Grafik
3862 DF             RST    18H           :Beide Adresse identisch ?
3863 2809           JR     Z,386EH       :Ja: Hochauflösende Grafik nicht
                                         ;erlaubt. (MOD SEL wurde während
                                         ;des Einschaltens festgehalten.
                                         ;Siehe S.20 im Handbuch)
3865 210048         LD     HL,4800H      :HL = Startadresse des Grafik-
                                         ;speichers
3868 71             LD     (HL),C        :Byte ablegen
3869 01FF0F         LD     BC,0FFFH      :BC = Zähler für 4095 Bytes
386C EDB0           LDIR                 :FCLS durchführen
386E E1             POP    HL            :PTZ zurück
386F C9             RET

; CRTC auf LGR- oder FGR-Modus programmieren

3870 3A1C43         LD     A,(431CH)     :A = Letzter Wert des Ports 255
3873 E5             PUSH   HL            :PTZ retten
3874 21F042         LD     HL,42F0H      :HL = Tabellenstart für LGR-Modus
3877 CB6F           BIT    05H,A         :Bit 5 gesetzt ?
3879 2803           JR     Z,387EH       :Nein: LGR-Modus programmieren
387B 210043         LD     HL,4300H      :Ja: FGR-Modus programmieren
387E D3FF           OUT    (0FFH),A      :Portwert zurückschreiben
3880 321C43         LD     (431CH),A     :und abspeichern
3883 0610           LD     B,10H         :16 Werte programmieren
3885 0EFA           LD     C,0FAH        :C = Portadresse zur Registerwahl
3887 05             DEC    B             :Zähler -1
3888 ED41           OUT    (C),B         :Register anwählen
388A 04             INC    B             :Zähler +1
388B 0C             INC    C             :C = Portadresse zur Wertübergabe
388C EDA3           OUTI                 :Byte von (HL) nach Port C
                                         ;übergeben
388E 20F5           JR     NZ,3885H      :Nächstes Byte
3890 E1             POP    HL            :PTZ zurück
3891 C9             RET

; CRTC initialisieren (NBGRD, LGR) und Text ab HL ausgeben
; I: HL = Startadresse des auszugebenden Textes

3892 D9             EXX                  :Register retten
3893 3A1C43         LD     A,(431CH)     :A = letzter Portwert
3896 E6DB           AND    0DBH          :Bits 2 und 5 löschen
                                         ;(NBGRD und LGR)
3898 CD7338         CALL   3873H         :CRTC neu programmieren
389B D9             EXX                  :Register zurück
389C CDA728         CALL   28A7H         :Text ab (HL) ausgeben
389F C9             RET
```

```
; UPRO für CONT (siehe 1DF2H)
; CRTC auf letzten Wert programmieren und letzte ZN als aktuelle ZN abspeichern

  38A0 D9               EXX                  ;Register retten
  38A1 CD7038           CALL    3870H        ;CRTC programmiren
  38A4 D9               EXX                  ;Register zurück
  38A5 22A240           LD      (40A2H),HL   ;aktuelle ZN erneuern
  38A8 C9               RET

; FGR

  38A9 3A1C43           LD      A,(431CH)    ;A = Letzter Portwert
  38AC CBEF             SET     05H,A        ;Bit 5 für FGR setzen
  38AE 18C3             JR      3873H        ;und CRTC programmieren

; LGR                           -

  38B0 3A1C43           LD      A,(431CH)    ;A = Letzter Portwert
  38B3 CBAF             RES     05H,A        ;Bit 5 für LGR löschen
  38B5 18BC             JR      3873H        ;und CRTC programmieren

; Alte BGRD-Routine (wurde jetzt durch BGRD n ab 3FE4H ersetzt)

  38B7 0604             LD      B,04H
  38B9 1802             JR      38BDH

; NBGRD

  38BB 0600             LD      B,00H        ;BGRD-Wert auf 0 setzen
  38BD 3A1C43           LD      A,(431CH)    ;Letzten Portwert holen
  38C0 E6FB             AND     0FBH         ;BGRD-Bits löschen
  38C2 B0               OR      B            ;Neuen Wert einmaskieren
  38C3 321C43           LD      (431CH),A    ;abspeichern
  38C6 D3FF             OUT     (0FFH),A     ;und neuen BGRD erzeugen
  38C8 C9               RET

; COLOUR

  38C9 CDC23F           CALL    3FC2H        ;Wert nach A holen
  38CC FE10             CP      10H          ;Wert > 15 ?
  38CE D24A1E           JP      NC,1E4AH     ;Ja: FC-Error
  38D1 322340           LD      (4023H),A    ;Nein: Wert abspeichern
  38D4 C9               RET

; FCOLOUR
; Das 'FCOLOUR'-Token wird mit 3 Bytes abgespeichert ! (FFH, 81H, 52H)
; Aufgrund eines Fehlers im alten ROM wurde nur das Keyword 'FCOLOU' erkannt,
; das fehlende 'R' musste extra abgespeichert werden.

  38D5 CF               RST     08H          ;Folgt ein 'R' ?
  38D6 52               DEFB    'R'
  38D7 CDC23F           CALL    3FC2H        ;Wert nach A holen
  38DA FE04             CP      04H          ;Wert > 3 ?
  38DC 30F0             JR      NC,38CEH     ;Ja: FC-Error
  38DE 321343           LD      (4313H),A    ;Nein: Wert abspeichern
  38E1 C9               RET
```

```
; UPRO für die Zwischencodeerzeugung (siehe 1C15H)
; Beim Erreichen des Endes der normalen Keywordtabelle.
; Colour-Keywords abfragen
; I: A  = Aktuelles Tabellenzeichen
;    B  = Tokenzähler
;    C  = Aktuelles Textzeichen
;    DE = Textzeiger
;    HL = Tabellenzeiger

  38E2 E67F        AND   7FH        ;Tabellenende erreicht ?
  38E4 C0          RET   NZ         ;Nein: Zurück
  38E5 EB          EX    DE.HL      ;HL = Textzeiger
  38E6 112F39      LD    DE.392FH   ;DE = Tabellenzeiger auf Colour-
                                    ;Keyword-Tabelle
  38E9 C5          PUSH  BC         ;Tokenzähler retten
  38EA 067F        LD    B.7FH      ;Neuen Tokenzähler setzen
  38EC 7E          LD    A.(HL)     ;A = Aktuelles Textzeichen
  38ED FE61        CP    61H        ;Ist A Kleinbuchstabe ?
  38EF 3806        JR    C.38F7H    ;Nein: ok
  38F1 FE7B        CP    7BH        ;Kleinbuchstabe ?
  38F3 3002        JR    NC.38F7H   ;Nein: ok
  38F5 E65F        AND   5FH        ;Ja: In Großbuchstaben umwandeln
  38F7 4E          LD    C.(HL)     ;C = Aktuelles Textzeichen
  38F8 EB          EX    DE.HL      ;HL = Tabellenzeiger
  38F9 23          INC   HL         ;Tabellenzeiger +1
  38FA B6          OR    (HL)       ;Nächstes Keyword erreicht ?
  38FB F2F938      JP    P.38F9H    ;Nein: Zeiger bis zum nächsten
                                    ;Keyword erhöhen
  38FE 04          INC   B          ;Tokenzähler +1
  38FF 7E          LD    A.(HL)     ;A = Tabellenzeichen
  3900 E67F        AND   7FH        ;Tabellenende erreicht ?
  3902 2829        JR    Z.392DH    ;Ja: Rücksprung
  3904 B9          CP    C          ;Nein: Tabellenzeichen mit Text-
                                    ;zeichen vergleichen
  3905 20F2        JR    NZ.38F9H   ;Nächstes Keyword probieren wenn
                                    ;ungleich
  3907 EB          EX    DE.HL      ;DE = Tabellenzeiger
  3908 E5          PUSH  HL         ;Textzeiger retten
  3909 13          INC   DE         ;Tabellenzeiger +1
  390A 1A          LD    A.(DE)     ;nächstes Tabellenzeichen holen
  390B B7          OR    A          ;nächstes Keyword erreicht ?
  390C FA1E39      JP    M.391EH    ;Ja: Alle Zeichen stimmten über-
                                    ;ein
  390F 4F          LD    C.A        ;Nein: C = Tabellenzeichen
  3910 23          INC   HL         ;Textzeiger +1
  3911 7E          LD    A.(HL)     ;Textzeichen holen
  3912 FE61        CP    61H        ;in Großschrift umwandeln
  3914 3802        JR    C.3918H
  3916 E65F        AND   5FH
  3918 B9          CP    C          ;und mit Tabellenzeichen ver-
                                    ;gleichen
  3919 28EE        JR    Z.3909H    ;Nächstes Zeichen wenn indentisch
  391B E1          POP   HL         ;Alten Textzeiger zurück
  391C 18D9        JR    38F7H      ;und nächstes Keyword vergleichen
```

```
; Keyword gefunden

   391E F1              POP    AF         :Textzeiger
   391F F1              POP    AF         ;alten Tokenzähler
   3920 F1              POP    AF         :RET-Adr nach 1C18H
   3921 D1              POP    DE         ;und RET-Adr nach 1C3DH löschen
   3922 D1              POP    DE         :Zähler der Zeichen pro Zeile
                                          ;zurück
   3923 E3              EX     (SP).HL    :Textzeiger retten, Bufferzeiger
                                          ;zurück
   3924 36FF            LD     (HL),0FFH  ;FFH als Colour-Keyword Kennung
                                          ;im Buffer ablegen
   3926 78              LD     A,B        :A = Token des gefundenen
                                          :Keywords
   3927 42              LD     B,D        :BC = DE
   3928 4B              LD     C,E -
   3929 D1              POP    DE         ;Textzeiger zurück nach DE
   392A C3573D          JP     3D57H      ;und Token ablegen

; Ende der Keywordtabelle erreicht, kein Keyword erkannt

   392D C1              POP    BC         ;alten Zähler zurück
   392E F1              POP    AF         :RET-Adr nach 1C18H löschen
   392F C9              RET               ;und RET nach 1C3DH ausführen

; Tabelle der Colour-Keywords in der Reihenfolge ihrer Tokenwerte
; Zur Trennung zwischen den Keywords ist jeweils das 7. Bit des ersten Zeichens
; auf '1' gesetzt

   3930 C3 4F 4C 4F 55 52                 :COLOUR
   3936 C6 43 4F 4C 4F 55                 :FCOLOU (das 'R' fehlt !!)
   393C CB 45 59 50 41 44                 :KEYPAD
   3942 CA 4F 59                          :JOY
   3945 D0 4C 4F 54                       :PLOT
   3949 C6 47 52                          :FGR
   394C CC 47 52                          :LGR
   394F C6 43 4C 53                       :FCLS
   3953 D0 4C 41 59                       :PLAY
   3957 C3 49 52 43 4C 45                 :CIRCLE
   395D D3 43 41 4C 45                    :SCALE
   3962 D3 48 41 50 45                    :SHAPE
   3967 CE 53 48 41 50 45                 :NSHAPE
   396D D8 53 48 41 50 45                 :XSHAPE
   3973 D0 41 49 4E 54                    :PAINT
   3978 C3 50 4F 49 4E 54                 :CPOINT
   397E CE 50 4C 4F 54                    :NPLOT
   3983 D3 4F 55 4E 44                    :SOUND
   3988 C3 48 41 52                       :CHAR
   398C D2 45 4E 55 4D                    :RENUM
   3991 D3 57 41 50                       :SWAP
   3995 C6 4B 45 59                       :FKEY
   3999 C3 41 4C 4C                       :CALL
   399D D6 45 52 49 46 59                 :VERIFY
   39A3 C2 47 52 44                       :BGRD
   39A7 CE 42 47 52 44                    :NBGRD
   39AC 80                                :Ende der Keywordtabelle
```

```
; UPRO für LIST (siehe 2BA9H)
; Start der Keywordtabelle festlegen

  39AD FE80          CP     80H          ;Tokenwert = FFH ? (FFH - 7FH)
  39AF 215016        LD     HL,1650H     ;HL = Startadresse der Basic
                                         ;Keywordtabelle
  39B2 C0            RET    NZ           ;Nein: HL ist gesuchte Adresse
  39B3 E1            POP    HL           ;RET-Adr nach HL
  39B4 E3            EX     (SP),HL      ;RET-Adr zurück,
                                         ;HL = Zeilenpointer
  39B5 7E            LD     A,(HL)       ;A = nächstes Zeichen
  39B6 D67F          SUB    7FH          ;Ist es ein Token ?
  39B8 5F            LD     E,A          ;E = Zeichen - 7FH
  39B9 23            INC    HL           ;Zeiger +1
  39BA E3            EX     (SP),HL      ;Zeiger retten, RET-Adr nach HL
  39BB E5            PUSH   HL           ;RET-Adr zurück ins Stack
  39BC 2A8C43        LD     HL,(438CH)   ;HL = Startadresse der Colour
                                         ;Keywordtabelle
  39BF C9            RET

; UPRO für die Programmschleife (siehe 1D67H)
; Startadresse der Sprungadressentabelle festlegen

  39C0 FE7F          CP     7FH          ;Colour-Token gefunden ?
  39C2 2808          JR     Z,39CCH      ;Ja: weiter bei 39CCH
  39C4 FE3C          CP     3CH          ;Befehl gefunden ?
  39C6 D2E72A        JP     NC,2AE7H     ;Nein: weiter bei 2AE7H
  39C9 C36A1D        JP     1D6AH        ;Ja: Befehlsroutine ausführen

; Colour-Token gefunden

  39CC 23            INC    HL           ;PTZ +1
  39CD 7E            LD     A,(HL)       ;Eigentliches Token holen
  39CE D680          SUB    80H          ;80H abziehen
  39D0 07            RLCA                ;*2 = Tabellenoffset
  39D1 4F            LD     C,A          ;C = (Tokenwert - 80H) * 2
  39D2 0600          LD     B,00H        ;B = 0
  39D4 EB            EX     DE,HL        ;DE = PTZ
  39D5 2A8E43        LD     HL,(438EH)   ;HL = Startadresse der Colour-
                                         ;Sprungadressentabelle
  39D8 C3721D        JP     1D72H        ;Adresse holen und Routine
                                         ;ausführen
```

: Sprungadressentabelle der Colour-Befehle

```
 39DB C9 38                                :COLOUR   38C9H
 39DD D5 38                                :FCOLOUR  38D5H
 39DF 97 19                                :KEYPAD   1997H (= SN-Error !)
 39E1 52 36                                :JOY      3652H
 39E3 C1 3B                                :PLOT     3BC1H
 39E5 A9 38                                :FGR      38A9H
 39E7 B0 38                                :LGR      38B0H
 39E9 83 3C                                :FCLS     3C83H
 39EB 61 3D                                :PLAY     3D61H
 39ED F8 3A                                :CIRCLE   3AF8H
 39EF F1 3A                                :SCALE    3AF1H
 39F1 DD 3C                                :SHAPE    3CDDH
 39F3 D8 3C                                :NSHAPE   3CD8H
 39F5 D3 3C                                :XSHAPE   3CD3H
 39F7 38 3E                                :PAINT    3E38H
 39F9 97 19                                :CPOINT   1997H (= SN-Error !)
 39FB BE 3B                                :NPLOT    3BBEH
 39FD 95 3F                                :SOUND    3F95H
 39FF A8 3F                                :CHAR     3FA8H
 3A01 B6 31                                :RENUM    31B6H
 3A03 D3 36                                :SWAP     36D3H
 3A05 66 34                                :FKEY     3466H
 3A07 5A 35                                :CALL     355AH
 3A09 33 3F                                :VERIFY   3F33H
 3A0B E4 3F                                :BGRD     3FE4H
 3A0D BB 38                                :NBGRD    38BBH
```

: KEYPAD

```
 3A0F 7E        LD     A,(HL)      :Nächstes Zeichen holen
 3A10 23        INC    HL          :PTZ +1
 3A11 E5        PUSH   HL          :PTZ retten
 3A12 FE31      CP     31H         :'1' gefunden ?
 3A14 CAD83A    JP     Z,3AD8H     :Ja: weiter bei 3AD8H
 3A17 FE32      CP     32H         :'2' gefunden ?
 3A19 CADC3A    JP     Z,3ADCH     :Ja: weiter bei 3ADCH
 3A1C C3C33A    JP     3AC3H       :Nein: KEYPAD (n) ?
```

```
; JOY

  3A1F 7E            LD      A,(HL)       ;A = Nächstes Zeichen
  3A20 FE28          CP      28H          ;Klammer auf '(' ?
  3A22 2826          JR      Z,3A4AH      ;Ja: weiter bei 3A4AH
  3A24 1604          LD      D,04H        ;D = 4
  3A26 FE32          CP      32H          ;'2' gefunden ?
  3A28 2807          JR      Z,3A31H      ;Ja: weiter bei 3A31H
  3A2A CB3A          SRL     D            ;Nein: D = 2 (4/2)
  3A2C FE31          CP      31H          ;'1' gefunden ?
  3A2E C29719        JP      NZ,1997H     ;Nein: SN-Error
  3A31 D7            RST     10H          ;Ja: Nächstes Zeichen holen
  3A32 FE59          CP      59H          ;'Y' gefunden ?
  3A34 2805          JR      Z,3A3BH      ;Ja: weiter bei 3A3BH
  3A36 15            DEC     D            ;Nein: D -1
  3A37 FE58          CP      58H -        ;'X' gefunden ?
  3A39 20F3          JR      NZ,3A2EH     ;Nein: SN-Error
  3A3B 23            INC     HL           ;PTZ +1
  3A3C E5            PUSH    HL           ;PTZ retten
  3A3D CD5E3A        CALL    3A5EH        ;Joystickwert holen
  3A40 E63F          AND     3FH          ;Wert auf 0 - 63 normieren
  3A42 3C            INC     A            ;+1 ergibt Wert von 1 bis 64
  3A43 1816          JR      3A5BH        ;Wert nach X als INT übergeben

; Unbenutzt

  3A45 18F4          JR      3A3BH        ;Joystickwert holen
  3A47 00            NOP                  ;--
  3A48 00            NOP
  3A49 00            NOP

; JOY (n)

  3A4A 23            INC     HL           ;PTZ +1
  3A4B CD1C2B        CALL    2B1CH        ;Wert holen
  3A4E FE08          CP      08H          ;Wert > 7 ?
  3A50 D24A1E        JP      NC,1E4AH     ;Ja: FC-Error
  3A53 57            LD      D,A          ;Nein: Wert nach D
  3A54 CF            RST     08H          ;Klammer geschlossen ?
  3A55 29            DEFB    ')'
  3A56 14            INC     D            ;D +1 (Wertebereich von 1 bis 8)
  3A57 E5            PUSH    HL           ;PTZ retten
  3A58 CD5E3A        CALL    3A5EH        ;Joystickwert holen
  3A5B C3723F        JP      3F72H        ;und nach X als INT übergeben
```

```
: Joystickwert ermitteln
: I: D = Parameter für anzusprechenden Joystick: (zwischen 1 und 8)
:        D = 1: JOY1X   D = 2: JOY1Y   D = 3: JOY2X   D = 4: JOY2Y
:        die restlichen Werte (5 bis 8) sprechen das zweite Paar
:        Joysticks an (bzw. 4 andere Analogeingänge)
: O: A = Ermittelter Wert

  3A5E CDA93A          CALL   3AA9H        :Port A zur Ausgabe, Port B zur
                                           :Eingabe vorbereiten
  3A61 AF              XOR    A            :A = 0
  3A62 1E80            LD     E,80H        :E.7 = 1
  3A64 B3              OR     E            :Das jeweils gesetzte Bit in E
                                           :auch in A setzen
  3A65 6F              LD     L,A          :L = A
  3A66 260E            LD     H,0EH        :Port A ansprechen
  3A68 CDB33A          CALL   3AB3H        :Wert in L nach Port A ausgeben
  3A6B 24              INC    H            :Port B ansprechen
  3A6C CDBB3A          CALL   3ABBH        :Wert aus Port B nach A holen
  3A6F D5              PUSH   DE           :Zähler (D) retten
  3A70 17              RLA                 :Gewünschtes Bit nach CY schieben
  3A71 15              DEC    D            :Gewünschtes Bit in CY ?
  3A72 20FC            JR     NZ,3A70H     :Nein: Weiterschieben
  3A74 D1              POP    DE           :Zähler zurück
  3A75 7D              LD     A,L          :Letzten ausgegebenen Wert nach A
  3A76 3803            JR     C,3A7BH      :War das gewünschte Bit = '1' ?
                                           :Ja: weiter bei 3A7BH
  3A78 7B              LD     A,E          :Nein: A = Bitmaske
  3A79 2F              CPL                 :Maske invertieren
  3A7A A5              AND    L            :und mit L verknüpfen
  3A7B CB3B            SRL    E            :Maske rechts schieben
  3A7D 30E5            JR     NC,3A64H     :CY=1 ? (Maske 8 mal geschoben ?)
                                           :Nein: nächsten Durchgang
  3A7F C9              RET

: Alte KEYPAD-Routine (jetzt ab 3A0FH)

  3A80 C3143F          JP     3F14H        :Früher: KEYPAD1
  3A83 00              NOP                 :--
  3A84 C3183F          JP     3F18H        :Früher: KEYPAD2
```

```
; Aktuelle KEYPAD-Routine
; I: D = FEH: KEYPAD1 ansprechen
;    D = F7H: KEYPAD2 ansprechen
; O: A = Tastenwert

  3A87 CDA93A        CALL   3AA9H       ;Port A zur Ausgabe, Port B zur
                                        ;Eingabe vorbereiten
  3A8A 1EE4          LD     E,0E4H      ;E = LSB der Startadresse der
                                        ;KEYPAD-Tabelle
  3A8C 0603          LD     B,03H       ;3 Spalten prüfen
  3A8E 6A            LD     L,D         ;L = Keypad-Adresse
  3A8F 260E          LD     H,0EH       ;Port A ansprechen
  3A91 CDB33A        CALL   3AB3H       ;und gewünschten Keypad
                                        ;adressieren
  3A94 24            INC    H           ;Port B ansprechen
  3A95 CDBB3A        CALL   3ABBH       ;und Wert holen
  3A98 0E04          LD     C,04H       ;4 Zeilen prüfen
  3A9A 1F            RRA                ;nächstes Bit nach CY schieben
  3A9B 3008          JR     NC,3AA5H    ;Sprung wenn Taste gedrückt
  3A9D 1C            INC    E           ;Tabellenoffset +1
  3A9E 0D            DEC    C           ;Zeilenzähler -1
  3A9F 20F9          JR     NZ,3A9AH    ;Nächste Zeile prüfen
  3AA1 CB02          RLC    D           ;Nächste Spalte ansprechen
  3AA3 10E9          DJNZ   3A8EH       ;Weiter bis alle 3 Spalten über-
                                        ;prüft wurden
  3AA5 163A          LD     D,3AH       ;D = MSB der Startadresse der
                                        ;KEYPAD-Tabelle
  3AA7 1A            LD     A,(DE)      ;Wert aus Tabelle holen
  3AA8 C9            RET

; Port A zur Ausgabe und Port B zur Eingabe vorbereiten

  3AA9 2607          LD     H,07H       ;Register 7 des PSG ansprechen
  3AAB CDBB3A        CALL   3ABBH       ;Wert holen
  3AAE E63F          AND    3FH         ;Bits 6 und 7 löschen
  3AB0 F640          OR     40H         ;Bit 6 auf '1' und Bit 7 auf '0'
                                        ;setzen

; Wert in A zum PSG-Register H übergeben

  3AB2 6F            LD     L,A         ;L = neuer Wert
  3AB3 0EF8          LD     C,0F8H      ;C = Port für Registerwahl
  3AB5 ED61          OUT    (C),H       ;Register H anwählen
  3AB7 0C            INC    C           ;C = Port zur Werteübergabe
  3AB8 ED69          OUT    (C),L       ;Wert übergeben
  3ABA C9            RET

; Wert des PSG-Registers H holen

  3ABB 0EF8          LD     C,0F8H      ;C = Port für Registerwahl
  3ABD ED61          OUT    (C),H       ;Register H anwählen
  3ABF 0C            INC    C           ;C = Port zur Werteübergabe
  3AC0 ED78          IN     A,(C)       ;Wert holen
  3AC2 C9            RET
```

```
: KEYPAD (n)

  3AC3 E1            POP    HL          :PTZ zurück
  3AC4 2B            DEC    HL          :PTZ -1
  3AC5 CF            RST    08H         :Klammer auf ?
  3AC6 28            DEFB   '('         :Nein: SN-Error
  3AC7 CDC23F        CALL   3FC2H       :Wert -1 holen
  3ACA FE02          CP     02H         :Wert > 2 ?
  3ACC D24A1E        JP     NC,1E4AH    :Ja: FC-Error
  3ACF F5            PUSH   AF          :Nein: Wert retten
  3AD0 CF            RST    08H         :Klammer zu ?
  3AD1 29            DEFB   ')'
  3AD2 F1            POP    AF          :Wert zurück
  3AD3 E5            PUSH   HL          :PTZ retten
  3AD4 FE01          CP     01H         :Wert = 2 ? (A = Wert - 1)
  3AD6 2804          JR     Z,3ADCH     :Ja: KEYPAD2 ausführen

: KEYPAD1

  3AD8 16FE          LD     D,0FEH      :D = Keypad1-'Adresse'
  3ADA 1802          JR     3ADEH       :weiter bei 3ADEH

: KEYPAD2

  3ADC 16F7          LD     D,0F7H      :D = Keypad2-'Adresse'
  3ADE CD873A        CALL   3A87H       :Keypadwert holen
  3AE1 C3723F        JP     3F72H       :und nach X als INT übergeben

: KEYPAD-Tabelle
: Zur schnelleren Berechnung der Tastenwerte, werden diese über eine Tabelle
: ermittelt (Jeweils 3 Spalten a 4 Zeilen)

  3AE4 03
  3AE5 06
  3AE6 09
  3AE7 0C

  3AE8 02
  3AE9 05
  3AEA 08
  3AEB 0A

  3AEC 01
  3AED 04
  3AEE 07
  3AEF 0B

  3AF0 00                                  :Keine Taste gedrückt
```

```
: SCALE

  3AF1 CD1C2B          CALL    2B1CH          ;Wert holen
  3AF4 321443          LD      (4314H),A      ;und abspeichern
  3AF7 C9              RET

: CIRCLE

  3AF8 CD1C2B          CALL    2B1CH          ;X-Koordinate holen
  3AFB F5              PUSH    AF             ;und retten
  3AFC CF              RST     08H            ;Danach muß ',' stehen
  3AFD 2C              DEFB    ','
  3AFE CD1C2B          CALL    2B1CH          ;Y-Koordinate holen
  3B01 F5              PUSH    AF             ;und retten
  3B02 CF              RST     08H            ;Danach muß ',' stehen
  3B03 2C              DEFB    ','
  3B04 CD1C2B          CALL    2B1CH          ;Radius holen
  3B07 D1              POP     DE             ;Y-Koordinate zurück
  3B08 C1              POP     BC             ;X-Koordinate zurück
  3B09 4A              LD      C,D            ;B = X-, C = Y-Koordinate
  3B0A E5              PUSH    HL             ;PTZ retten
  3B0B 57              LD      D,A            ;Y-Abstand = Radius
  3B0C 1E00            LD      E,00H          ;X-Abstand = 0
  3B0E 2680            LD      H,80H          ;H = Schrittweitenoffset

: Nächste acht Punkte plotten

  3B10 FA523B          JP      M,3B52H        ;Fertig wenn Y-Abstand kleiner
                                              ;als der X-Abstand ist
                                              ;(Kreis geschlossen)
  3B13 CD7A3B          CALL    3B7AH          ;4 Punkte plotten und X- und
                                              ;Y-Abstände vertauschen
  3B16 CD7A3B          CALL    3B7AH          ;4 Punkte plotten und Abstände
                                              ;wieder zurück
  3B19 CD5E3B          CALL    3B5EH          ;Abstände negieren
  3B1C C5              PUSH    BC             ;Register retten
  3B1D D5              PUSH    DE
  3B1E E5              PUSH    HL
  3B1F 2E80            LD      L,80H          ;L. 7 = 1 (Dadurch ergibt sich
                                              ;beim ersten ADD HL,HL ein
                                              ;Überlauf nach H.0)
  3B21 63              LD      H,E            ;H = X-Abstand
  3B22 0608            LD      B,08H          ;8 Bits verrechnen
  3B24 1E00            LD      E,00H          ;E = 0, D = Y-Abstand
```

```
: Nächste Schrittweite für den Y-Abstand errechnen

  3B26 29            ADD    HL,HL       ;nächstes Bit von H nach CY
                                        ;schieben
  3B27 ED52          SBC    HL,DE       ;HL = HL - DE - CY
  3B29 3803          JR     C,3B2EH     ;Sprung bei Unterlauf
                                        ;(Da E = 0 ist kann ein Unterlauf
                                        ;nur erfolgen wenn H < D ist)
  3B2B 23            INC    HL          ;HL +1 (entspricht L +1)
  3B2C 1801          JR     3B2FH       ;Nächster Durchgang
  3B2E 19            ADD    HL,DE       ;Subtraktion zurücknehmen
  3B2F 10F5          DJNZ   3B26H       ;Nächster Durchgang

  3B31 7D            LD     A,L         ;A = L
  3B32 E1            POP    HL          ;Register zurück
  3B33 D1            POP    DE  -
  3B34 C1            POP    BC
  3B35 84            ADD    A,H         ;A = A + Schrittweitenoffset
                                        ;CY = 1 falls A + H > 255
  3B36 67            LD     H,A         ;H = neuer Schrittweitenoffset
  3B37 7A            LD     A,D         ;A = Y-Abstand
  3B38 2E00          LD     L,00H       ;L = 0
  3B3A 9D            SBC    A,L         ;A = Y-Abstand - CY
  3B3B 57            LD     D,A         ;D = neuer Y-Abstand
  3B3C 1C            INC    E           ;X-Abstand +1
  3B3D 7A            LD     A,D         ;A = Y-Abstand
  3B3E BB            CP     E           ;X-Abstand > Y-Abstand ?
  3B3F C3103B        JP     3B10H       ;weiter bei 3B10H

: PLOT  B + E , C + D

  3B42 C5            PUSH   BC          ;Register retten
  3B43 D5            PUSH   DE
  3B44 E5            PUSH   HL
  3B45 7B            LD     A,E         ;A = X-Abstand
  3B46 80            ADD    A,B         ;+ Mittelpunktskoordinate
  3B47 6F            LD     L,A         ;ergibt X-Koordinate
  3B48 7A            LD     A,D         ;A = Y-Abstand
  3B49 81            ADD    A,C         ;+ Mittelpunktskoordinate
  3B4A 67            LD     H,A         ;ergibt Y-Koordinate
  3B4B CD8A3B        CALL   3B8AH       ;PLOT  L . H
  3B4E E1            POP    HL          ;Register zurück
  3B4F D1            POP    DE
  3B50 C1            POP    BC
  3B51 C9            RET

: CIRCLE beenden

  3B52 E1            POP    HL          ;PTZ zurück
  3B53 C9            RET
```

```
; X-Abstand negieren ( E = -E )

  3B54 7B          LD     A,E        ;A = E
  3B55 ED44        NEG               ;A = -A
  3B57 5F          LD     E,A        ;neuen Wert zurück nach E
  3B58 C9          RET

; Y-Abstand negieren ( D = -D )

  3B59 7A          LD     A,D        ;A = D
  3B5A ED44        NEG               ;A = -A
  3B5C 57          LD     D,A        ;neuen Wert zurück nach D
  3B5D C9          RET

; X- und Y-Abstand negieren

  3B5E CD543B      CALL   3B54H      ;X-Abstand negieren
  3B61 CD593B      CALL   3B59H      ;Y-Abstand negieren
  3B64 C9          RET

; PLOT  B - E , C + D

  3B65 CD543B      CALL   3B54H      ;X-Abstand negieren
  3B68 CD423B      CALL   3B42H      ;PLOT  B + E , C + D
  3B6B C9          RET

; PLOT B + E , C - D

  3B6C CD593B      CALL   3B59H      ;Y-Abstand negieren
  3B6F CD423B      CALL   3B42H      ;PLOT  B + E , C + D
  3B72 C9          RET

; X- und Y-Abstand vertauschen

  3B73 7B          LD     A,E        ;L = E
  3B74 6F          LD     L,A
  3B75 7A          LD     A,D        ;E = D
  3B76 5F          LD     E,A
  3B77 7D          LD     A,L        ;D = L
  3B78 57          LD     D,A
  3B79 C9          RET

; 4 Punkte plotten und X- und Y-Abstand vertauschen

  3B7A CD423B      CALL   3B42H      ;PLOT  B + E , C + D
                                     ;(entspricht rechts unten)
  3B7D CD653B      CALL   3B65H      ;PLOT  B - E , C + D
                                     ;(entspricht links unten)
  3B80 CD6C3B      CALL   3B6CH      ;PLOT  B - E , C - D
                                     ;(entspricht links oben)
  3B83 CD653B      CALL   3B65H      ;PLOT  B + E , C - D
                                     ;(entspricht rechts oben)
  3B86 CD733B      CALL   3B73H      ;Abstände vertauschen
                                     ;(für 2. Aufruf, siehe 3B13H ff)
  3B89 C9          RET
```

```
; UPRO für PLOT, PAINT, SHAPE und CIRCLE
; PLOT L , H (AF,BC,DE,HL)
; I:  L       = X-Koordinate
;      H      = Y-Koordinate
;    (4313H) = Farbcode (entspricht FCOLOUR)

  3B8A 3A1343        LD     A,(4313H)    ;FCOLOUR-Wert holen
  3B8D E603          AND    03H          ;Auf Werte zwischen 0 und 3
                                         ;normieren

; PLOT L , H
; wie 3B8AH aber A = Farbcode

  3B8F 4F            LD     C,A          ;C = Farbcode
  3B90 3E9F          LD     A,9FH        ;A = max. X-Wert (159)
  3B92 BD            CP     L            ;max. X-Wert überschritten ?
  3B93 D8            RET    C            ;Ja: Fertig
  3B94 3E65          LD     A,65H        ;A = max. Y-Wert (101)
  3B96 BC            CP     H            ;max. Y-Wert überschritten ?
  3B97 D8            RET    C            ;Ja: Fertig

  3B98 7D            LD     A,L          ;A = X-Wert
  3B99 6C            LD     L,H          ;L = Y-Wert
  3B9A 2600          LD     H,00H        ;HL = Y-Wert
  3B9C 54            LD     D,H          ;DE = HL
  3B9D 5D            LD     E,L
  3B9E 29            ADD    HL,HL        ;HL = Y-Wert *  2
  3B9F 29            ADD    HL,HL        ;HL = Y-Wert *  4
  3BA0 19            ADD    HL,DE        ;HL = Y-Wert *  5
  3BA1 29            ADD    HL,HL        ;HL = Y-Wert * 10
  3BA2 29            ADD    HL,HL        ;HL = Y-Wert * 20
  3BA3 29            ADD    HL,HL        ;HL = Y-Wert * 40
                                         ;(Eine Zeile des Grafikbild-
                                         ;schirms entspricht 40 Bytes im
                                         ;Grafikspeicher)
  3BA4 5F            LD     E,A          ;E = X-Wert
  3BA5 CB3B          SRL    E            ;E = X-Wert * 2
  3BA7 CB3B          SRL    E            ;E = X-Wert * 4
                                         ;(Ein Byte des Grafikspeichers
                                         ;entspricht 4 X-Positionen)
  3BA9 1648          LD     D,48H        ;DE = Startadresse des Grafik-
                                         ;speichers + X-Wert * 4
  3BAB 19            ADD    HL,DE        ;Zum Y-Wert*40 hinzuaddiert
                                         ;ergibt dies die Adresse des zu
                                         ;ändernden Bytes im Grafik-
                                         ;speicher
  3BAC E603          AND    03H          ;Position des zu setzenden Punkts
  3BAE 3C            INC    A            ;im Byte ermitteln
  3BAF 47            LD     B,A          ;B = X MOD 4
  3BB0 3EFC          LD     A,0FCH       ;A = 11111100
                                         ;(2 Bits = 1 Grafikpunkt)
  3BB2 0F            RRCA                ;Die beiden 0-Bits an die
  3BB3 0F            RRCA                ;gewünschte Stelle schieben
  3BB4 CB09          RRC    C            ;Farbcode mitschieben
  3BB6 CB09          RRC    C
  3BB8 10F8          DJNZ   3BB2H        ;weiter bis Zähler = 0
```

```
3BBA A6          AND   (HL)         :Byte des gewünschten Grafik-
                                    :punktes holen
3BBB B1          OR    C            :und neuen Farbcode setzen
3BBC 77          LD    (HL).A       :neues Byte zurückschreiben
3BBD C9          RET

; NPLOT

3BBE 0600        LD    B.00H        :B = Maske für Farbe löschen
3BC0 3A0603      LD    A.(0306H)    :--

; PLOT

*3BC1  0603      LD    B.03H        :B = Maske für Farbe lassen
3BC3 3A1343      LD    A.(4313H)    :A = FCOLOUR-Code
3BC6 F5          PUSH  AF           :FCOLOUR-Code retten
3BC7 A0          AND   B            :FCOLOUR-Code maskieren
3BC8 321343      LD    (4313H).A    :und für PLOT L.H abspeichern
3BCB CD7B3C      CALL  3C7BH        :'TO'-Token angegeben ?
3BCE 382F        JR    C.3BFFH      :Nein: weiter bei 3BFFH
3BD0 3A1543      LD    A.(4315H)    :Ja: letzten X-Wert holen
3BD3 F5          PUSH  AF           :und retten
3BD4 3A1643      LD    A.(4316H)    :Letzten Y-Wert holen
3BD7 F5          PUSH  AF           :und retten
3BD8 CD1C2B      CALL  2B1CH        :nächsten X-Wert holen
3BDB 321543      LD    (4315H).A    :abspeichern
3BDE F5          PUSH  AF           :und retten
3BDF CF          RST   08H          :Jetzt muß '.' folgen
3BE0 2C          DEFB  '.'
3BE1 CD1C2B      CALL  2B1CH        :nächsten Y-Wert holen
3BE4 321643      LD    (4316H).A    :und abspeichern
3BE7 D9          EXX                :Registersatz tauschen
3BE8 6F          LD    L.A          :L' = Y2
3BE9 D1          POP   DE           :X-Wert zurück
3BEA 62          LD    H.D          :H' = X2
3BEB C1          POP   BC           :Letzten Y-Wert zurück
3BEC D1          POP   DE           :Letzten X-Wert zurück
3BED 58          LD    E.B          :E' = Y1. D' = X1
3BEE D9          EXX                :Registersatz tauschen
3BEF E5          PUSH  HL           :PTZ retten
3BF0 D9          EXX                :Registersatz tauschen
3BF1 CD1F3C      CALL  3C1FH        :PLOT  D . E  TO  H . L
3BF4 E1          POP   HL           :PTZ zurück
3BF5 CD7B3C      CALL  3C7BH        :'TO' angegeben ?
3BF8 28D6        JR    Z.3BD0H      :Ja: nächstes X.Y-Paar holen
3BFA F1          POP   AF           :Nein: Alten FCOLOUR-Code
3BFB 321343      LD    (4313H).A    :zurückschreiben
3BFE C9          RET
```

```
: Neue X,Y-Werte holen ('TO' nicht am Anfang)

  3BFF CD1C2B        CALL    2B1CH        ;X-Wert holen
  3C02 F5            PUSH    AF           ;und retten
  3C03 CF            RST     08H          ;Jetzt muß ',' folgen
  3C04 2C            DEFB    ','
  3C05 CD1C2B        CALL    2B1CH        ;Y-Wert holen
  3C08 F5            PUSH    AF           ;und retten
  3C09 CD7B3C        CALL    3C7BH        ;'TO' angegeben ?
  3C0C 30CA          JR      NC,3BD8H     ;Ja: zweites X,Y-Paar holen
  3C0E F1            POP     AF           ;Nein: Y-Wert zurück
  3C0F 321643        LD      (4316H),A    ;und abspeichern
  3C12 57            LD      D,A          ;D = Y-Wert
  3C13 F1            POP     AF           ;X-Wert zurück
  3C14 321543        LD      (4315H),A    ;und abspeichern
  3C17 5F            LD      E,A          ;E = X-Wert
  3C18 EB            EX      DE,HL        ;DE = PTZ, H = Y-, L = X-Wert
  3C19 D5            PUSH    DE           ;PTZ retten
  3C1A CD8A3B        CALL    3B8AH        ;PLOT  L , H
  3C1D 18D5          JR      3BF4H        ;PTZ und alten FCOLOUR-Code
                                          ;zurück

: UPRO für PLOT
: PLOT  D , E  TO  H , L (AF,BC,DE,HL)
: I: D = X-Koordinate des Startpunkts
:    E = Y-Koordinate des Startpunkts
:    H = X-Koordinate des Endpunkts
:    L = Y-Koordinate des Endpunkts

  3C1F CDC63C        CALL    3CC6H        ;PLOT  H , L (Endpunkt plotten)
  3C22 DF            RST     18H          ;HL = DE ? (Start- und Endpunkt
                                          ;identisch ?)
  3C23 C8            RET     Z            ;Ja: Fertig
  3C24 00            NOP                  ;--
  3C25 00            NOP
  3C26 D5            PUSH    DE           ;X1,Y1 retten
  3C27 7B            LD      A,E          ;A = Y1
  3C28 95            SUB     L            ;A = Y1 - Y2 = Differenz der
                                          ;Y-Werte
  3C29 DC8B3C        CALL    C,3C8BH      ;Bei negativer Differenz
                                          ;A negieren und CY = 1 setzen
  3C2C CB19          RR      C            ;CY nach C.7 schieben
  3C2E CB39          SRL     C            ;C rechts schieben
  3C30 47            LD      B,A          ;B = Y-Diff (immer positiv !)
  3C31 CB39          SRL     C            ;C rechts schieben
  3C33 CB39          SRL     C            ;C rechts schieben
  3C35 7A            LD      A,D          ;A = X1
  3C36 94            SUB     H            ;-X2 = Differenz der X-Werte
  3C37 DC8B3C        CALL    C,3C8BH      ;Bei negativer Differenz
                                          ;A negieren und CY = 1 setzen
```

```
3C3A CB19          RR     C           ;CY nach C.7 schieben
3C3C 37            SCF                ;CY = 1
3C3D CB19          RR     C           ;C rechts schieben, C.7 = 1
3C3F B8            CP     B           ;X-Diff < Y-Diff ?
3C40 384D          JR     C,3C8FH     ;Ja: Die Half-Bytes von C ver-
                                      ;tauschen, E = X-Diff, D = Y-Diff
3C42 57            LD     D,A         ;Nein: D = X-Diff
3C43 78            LD     A,B
3C44 5F            LD     E,A         ;E = Y-Diff

; In D steht jetzt die größere Differenz und in E die kleinere
; Die beiden Half-Bytes von C geben die jeweilige Steigung an:
; (Das obere Half-Byte gilt für D, das untere für E)

3C45 C5            PUSH   BC          ;Steigung retten
3C46 E5            PUSH   HL -        ;X2,Y2 retten
3C47 7A            LD     A,D         ;A = Größere Differenz
3C48 4F            LD     C,A         ;C = A
3C49 3E00          LD     A,00H       ;A = 0
3C4B 57            LD     D,A         ;D = 0
3C4C 47            LD     B,A         ;B = 0 -> BC = Größere Diff
3C4D 67            LD     H,A         ;H = 0
3C4E 7B            LD     A,E         ;A = Kleinere Differenz
3C4F 6F            LD     L,A         ;L = A -> HL = Kleinere Diff
3C50 CB25          SLA    L           ;HL = HL * 2
3C52 CB14          RL     H           ;(HL ein Bit nach links)
3C54 ED42          SBC    HL,BC       ;HL = Kleinere Diff * 2 - Größere
                                      ;Differenz
3C56 CB21          SLA    C           ;BC = BC * 2 = Größere Diff * 2
3C58 CB10          RL     B           ;(BC ein Bit nach links)
3C5A CB23          SLA    E           ;E = Kleinere Differenz * 2
3C5C CB12          RL     D           ;D = Größere Differenz * 2
3C5E 7C            LD     A,H         ;A = H
3C5F D9            EXX                ;Registersatz tauschen
3C60 E1            POP    HL          ;X2,Y2 zurück
3C61 C1            POP    BC          ;Steigung zurück
3C62 D1            POP    DE          ;X1,Y1 zurück

; Nächsten Punkt errechnen und plotten

3C63 CB27          SLA    A           ;A.7 nach CY, CY = 1 ?
3C65 D4A03C        CALL   NC,3CA0H    ;Nein: X2 und Y2 entsprechend
                                      ;der Steigung der kleineren
                                      ;Differenz verändern
3C68 CDAF3C        CALL   3CAFH       ;X2 und Y2 entsprechend der
                                      ;Steigung der größeren Differenz
                                      ;verändern
```

```
3C6B CDC63C          CALL    3CC6H       :PLOT  H . L
3C6E 7A              LD      A.D         :A = X1
3C6F BC              CP      H           :X1 = X2 ?
3C70 2003            JR      NZ.3C75H    :Nein: nächsten Punkt setzen
3C72 7B              LD      A.E         :A = Y1
3C73 BD              CP      L           :Y1 = Y2 ?
3C74 C8              RET     Z           :Ja: Fertig
3C75 D9              EXX                 :Registersatz tauschen
3C76 19              ADD     HL.DE       :HL' = HL' + DE'
3C77 7C              LD      A.H         :A = H'
3C78 D9              EXX                 :Registersatz tauschen
3C79 18E8            JR      3C63H       :Nächsten Punkt setzen

; 'TO' angegeben ?

3C7B 7E              LD      A.(HL)      :A = nächstes Zeichen
3C7C FEBD            CP      0BDH        :'TO'-Token ?
3C7E 200D            JR      NZ.3C8DH    :Nein: CY = 1
3C80 23              INC     HL          :Ja: PTZ + 1
3C81 AF              XOR     A           :CY = 0
3C82 C9              RET

; FCLS

3C83 2B              DEC     HL          :PTZ -1
3C84 D7              RST     10H         :PTZ erhöhen
3C85 3E00            LD      A.00H       :A = 0 (Default-Farbwert)
3C87 C34A38          JP      384AH       :weiter bei 384AH

3C8A 00              NOP                 :--

; A = -A. CY = 1

3C8B ED44            NEG                 :A = -A
3C8D 37              SCF                 :CY = 1
3C8E C9              RET

; Halfbytes von C vertauschen. E = X-Diff. D = Y-Diff (siehe 3C40H)

3C8F CD973C          CALL    3C97H       :Halfbytes von C vertauschen
3C92 5F              LD      E.A         :E = X-Diff
3C93 78              LD      A.B         :A = Y-Diff
3C94 57              LD      D.A         :D = A
3C95 18AE            JR      3C45H       :zurück nach 3C45H

; Halfbytes von C vertauschen

3C97 CB09            RRC     C           :C viermal links rotieren
3C99 CB09            RRC     C
3C9B CB09            RRC     C
3C9D CB09            RRC     C
3C9F C9              RET
```

```
: X2 und Y2 entsprechend der Steigung der kleinere Differenz verändern

  3CA0 CD973C           CALL    3C97H        ;Halfbytes von C vertauschen
  3CA3 CDAF3C           CALL    3CAFH        ;X2 und Y2 verändern
  3CA6 CD973C           CALL    3C97H        ;Halfbytes von C vertauschen
  3CA9 D9               EXX                  ;Registersatz tauschen
  3CAA B7               OR      A            ;CY = 0
  3CAB ED42             SBC     HL,BC        ;HL' = HL' - BC'
  3CAD D9               EXX                  ;Registersatz vertauschen
  3CAE C9               RET

: X2 und Y2 entsprechend der Steigung der größeren Differenz verändern

  3CAF CB79             BIT     07H,C        ;Y2 verändern ?
  3CB1 CABD3C           JP      Z,3CBDH      ;Ja: weiter bei 3CBD
  3CB4 CB71             BIT     06H,C        ;Nein: X2 erhöhen ?
  3CB6 C2BB3C           JP      NZ,3CBBH     ;Nein: weiter bei 3CBBH
  3CB9 24               INC     H            ;Ja: X2 +1
  3CBA C9               RET
  3CBB 25               DEC     H            ;X2 -1
  3CBC C9               RET

  3CBD CB71             BIT     06H,C        ;Y2 erhöhen ?
  3CBF C2C43C           JP      NZ,3CC4H     ;Nein: weiter bei 3CC4H
  3CC2 2C               INC     L            ;Y2 +1
  3CC3 C9               RET
  3CC4 2D               DEC     L            ;Y2 -1
  3CC5 C9               RET

: UPRO für PLOT
: PLOT  H , L (AF)
: I: H = X-Koordinate
:    L = Y-Koordinate

  3CC6 C5               PUSH    BC           ;Register retten
  3CC7 D5               PUSH    DE
  3CC8 E5               PUSH    HL
  3CC9 7C               LD      A,H          ;H und L vertauschen
  3CCA 65               LD      H,L
  3CCB 6F               LD      L,A
  3CCC CD8A3B           CALL    3B8AH        ;PLOT  L , H
  3CCF E1               POP     HL           ;Register zurück
  3CD0 D1               POP     DE
  3CD1 C1               POP     BC
  3CD2 C9               RET
```

```
; XSHAPE

 3CD3 110303        LD      DE,0303H      ;DE = Maske für Farbe invertieren
 3CD6 1808          JR      3CE0H         ;weiter bei 3CE0H

; NSHAPE

 3CD8 110000        LD      DE,0000H      ;DE = Maske für Farbe löschen
 3CDB 1803          JR      3CE0H         ;weiter bei 3CE0H

; SHAPE

 3CDD 110003        LD      DE,0300H      ;DE = Maske für Farbe übernehmen
 3CE0 ED531743      LD      (4317H),DE    ;Maske abspeichern
 3CE4 2B            DEC     HL            ;PTZ -1
 3CE5 D7            RST     10H           ;PTZ erhöhen
 3CE6 CD1C2B        CALL    2B1CH         ;X-Koordinate holen
 3CE9 F5            PUSH    AF            ;und retten
 3CEA CF            RST     08H           ;Danach muß ',' stehen
 3CEB 2C            DEFB    ','
 3CEC CD1C2B        CALL    2B1CH         ;Y-Koordinate holen
 3CEF D1            POP     DE            ;X-Koordinate zurück nach D
 3CF0 5F            LD      E,A           ;E = Y-Koordinate
 3CF1 E5            PUSH    HL            ;PTZ retten
 3CF2 C3A235        JP      35A2H         ;HL mit der Startadresse der
                                          ;SHAPE-Tabelle laden
 3CF5 46            LD      B,(HL)        ;B = Länge der Tabelle
 3CF6 23            INC     HL            ;Zeiger +1
 3CF7 3A1443        LD      A,(4314H)     ;A = SCALE-Faktor
 3CFA B7            OR      A             ;SCALE 0 ?
 3CFB 284B          JR      Z,3D48H       ;Ja: Fertig
 3CFD 4F            LD      C,A           ;Nein: C = SACLE-Faktor
 3CFE 7E            LD      A,(HL)        ;A = Tabellenwert
 3CFF CB2F          SRA     A             ;Obere 4 Bits nach unten schieben
 3D01 CB2F          SRA     A
 3D03 CB2F          SRA     A
 3D05 CB2F          SRA     A
 3D07 08            EX      AF,AF'
 3D08 3E01          LD      A,01H         ;A' = 1 = Flag für die Be-
                                          ;arbeitung der ersten Linie
                                          ;(Jeder Tabellenwert entspricht
                                          ;zwei Linien)
 3D0A 08            EX      AF,AF'
 3D0B F5            PUSH    AF            ;Tabellenwert retten
 3D0C CB2F          SRA     A             ;Richtung vertikal ? (Bit0 =1)
 3D0E 383D          JR      C,3D4DH       ;Ja: weiter bei 3D4DH
 3D10 CB2F          SRA     A             ;Nein: Richtung links ? (Bit1 =1)
 3D12 3836          JR      C,3D4AH       ;Ja: weiter bei 3D4AH
 3D14 14            INC     D             ;Nein: X +1 (Richtung rechts)
```

```
3D15 C5            PUSH   BC          ;Register retten
3D16 D5            PUSH   DE
3D17 E5            PUSH   HL
3D18 6F            LD     L,A         ;Tabellenwert nach L
3D19 3A1343        LD     A,(4313H)   ;A = Aktueller FCOLOUR-Wert
3D1C F5            PUSH   AF          ;Aktuellen Wert retten
3D1D 7D            LD     A,L         ;Tabellenwert zurück nach A
3D1E 2A1743        LD     HL,(4317H)  ;HL = Maske für SHAPE, NSHAPE
                                      ;oder XSHAPE
3D21 A4            AND    H           ;Farbwert maskieren
3D22 AD            XOR    L
3D23 321343        LD     (4313H),A   ;Und als FCOLOUR-Wert für den
                                      ;PLOT-Aufruf abspeichern
3D26 6A            LD     L,D         ;L = X-Koordinate
3D27 63            LD     H,E         ;H = Y-Koordinate
3D28 CD8A3B        CALL   3B8AH       ;PLOT  L , H
3D2B F1            POP    AF          ;Aktuellen FCOLOUR-Wert
3D2C 321343        LD     (4313H),A   ;zurückschreiben
3D2F E1            POP    HL          ;Register zurück
3D30 D1            POP    DE
3D31 C1            POP    BC
3D32 F1            POP    AF          ;Tabellenwert zurück
3D33 0D            DEC    C           ;SCALE-Faktor > 1 ?
3D34 20D5          JR     NZ,3D0BH    ;Ja: Selben Tabellenwert noch-
                                      ;einmal (insgesamt also C-mal)
                                      ;bearbeiten
3D36 C5            PUSH   BC          ;Tabellenzähler retten
3D37 08            EX     AF,AF'
3D38 3D            DEC    A           ;Ist das Flag = 1 ?
3D39 47            LD     B,A         ;B = Flag -1
3D3A 08            EX     AF,AF'
3D3B 04            INC    B           ;B = 0 ?
3D3C 05            DEC    B           ;(Dann war das Flag = 1)
3D3D C1            POP    BC          ;Tabellenzähler zurück
3D3E 3A1443        LD     A,(4314H)   ;A = SCALE-Faktor
3D41 4F            LD     C,A         ;C = SCALE-Faktor
3D42 7E            LD     A,(HL)      ;A = Tabellenwert
3D43 28C6          JR     Z,3D0BH     ;Ja: 2. Linie bearbeiten
3D45 23            INC    HL          ;Sonst Tabellenzeiger erhöhen
3D46 10AF          DJNZ   3CF7H       ;und nächsten Wert bearbeiten
3D48 E1            POP    HL          ;PTZ zurück
3D49 C9            RET

: Richtung links

3D4A 15            DEC    D           ;X-Koordinate -1
3D4B 18C8          JR     3D15H       ;zurück zur SCALE-Routine

: Richtung vertikal

3D4D CB2F          SRA    A           ;Richtung oben ? (Bit1 =1)
3D4F 3803          JR     C,3D54H     ;Ja: weiter bei 3D45H
3D51 1C            INC    E           ;Nein: Y +1 (Richtung unten)
3D52 18C1          JR     3D15H       ;zurück zur SCALE-Routine
```

```
: Richtung oben

  3D54 1D            DEC    E          :Y-Koordinate -1
  3D55 18BE          JR     3D15H      :zurück zur SCALE-Routine

: UPRO für die Zwischencodeerzeugung (siehe 392AH)
: COLOUR-Token in die codierte Zeile übernehmen

  3D57 0C            INC    C          :Zeichenzähler +1
  3D58 23            INC    HL         :Bufferzeiger +1
                                       :(für das eingefügte FFH)
  3D59 EB            EX     DE.HL      :DE = Bufferzeiger (auf codierten
                                       :Text)
                                       :HL = Textzeiger (auf uncodierten
                                       :Text)
  3D5A 23            INC    HL         :Textzeiger +1
  3D5B 12            LD     (DE).A     :Token im Buffer ablegen
  3D5C 13            INC    DE         :Bufferzeiger +1
  3D5D 0C            INC    C          :Zeichenzähler +1
  3D5E C3CC1B        JP     1BCCH      :zurück zur Zwischencode-
                                       :erzeugung

: PLAY

  3D61 CF            RST    08H        :Klammer auf ?
  3D62 28            DEFB   '('
  3D63 CDC23F        CALL   3FC2H      :Kanalnummer holen
  3D66 FE03          CP     03H        :Kanal > 3 ?
                                       :(A ist Kanalnummer -1)
  3D68 D24A1E        JP     NC.1E4AH   :Ja: FC-Error
  3D6B F5            PUSH   AF         :Nein: Kanalnummer retten
  3D6C CF            RST    08H        :Jetzt muß ',' folgen
  3D6D 2C            DEFB   ','
  3D6E CDC43F        CALL   3FC4H      :Oktavnummer holen
  3D71 FE08          CP     08H        :Oktave > 8 ?
                                       :(A ist Oktavnummer -1)
  3D73 30F3          JR     NC.3D68H   :Ja: FC-Error
  3D75 3C            INC    A          :A = Oktavnummer
  3D76 F5            PUSH   AF         :Oktavnummer retten
  3D77 CF            RST    08H        :Jetzt muß ',' folgen
  3D78 2C            DEFB   ','
  3D79 CD1C2B        CALL   2B1CH      :Notenwert holen
  3D7C B7            OR     A          :Notenwert = 0 ?
  3D7D 283F          JR     Z.3DBEH    :Ja: weiter bei 3DBEH
  3D7F FE1E          CP     1EH        :Nein: Notenwert > 29 ?
  3D81 30E5          JR     NC.3D68H   :Ja: FC-Error
  3D83 CB27          SLA    A          :A = Notenwert * 2
  3D85 5F            LD     E.A        :DE = Notenwert * 2
  3D86 1600          LD     D.00H      :MSB von DE auf 0 setzen
  3D88 CF            RST    08H        :Jetzt muß ',' folgen
  3D89 2C            DEFB   ','
  3D8A E5            PUSH   HL         :PTZ retten
```

```
3D8B 21CF3D      LD      HL,3DCFH     ;HL = Startadresse der PLAY-
                                      ;Tabelle
3D8E 19          ADD     HL,DE        ;Notenwert * 2 addieren
3D8F 5E          LD      E,(HL)       ;und den 16-Bit Wert der
3D90 23          INC     HL           ;Tabelle entnehmen
3D91 56          LD      D,(HL)
3D92 E1          POP     HL           ;PTZ zurück
3D93 C1          POP     BC           ;B = Oktavnummer
3D94 05          DEC     B            ;Oktavnummer = 1 ?
3D95 2806        JR      Z,3D9DH      ;Ja: Werte lassen
3D97 CB3A        SRL     D            ;Nein: DE rechts schieben
3D99 CB1B        RR      E
3D9B 10FA        DJNZ    3D97H        ;DE = DE / (2 * Oktavwert)
3D9D F1          POP     AF           ;Kanalnummer -1 nach A
3D9E F5          PUSH    AF           ;und wieder retten
3D9F CB27        SLA     A            ;A = Kanalnummer * 2
3DA1 3C          INC     A            ; + 1
3DA2 CD2A3E      CALL    3E2AH        ;PSG-Register A-1 auf E, PSG-
                                      ;Register A auf D setzen
                                      ;(A zeigt auf die Register, die
                                      ;die Schwingungsdauer bestimmen)
3DA5 CD1C2B      CALL    2B1CH        ;Lautstärkenwert holen
3DA8 FE11        CP      11H          ;Lautstärke > 16 ?
3DAA 30BC        JR      NC,3D68H     ;Ja: FC-Error
3DAC D5          PUSH    DE           ;Nein: Lautstärke retten
3DAD 1E38        LD      E,38H        ;E = Programmierwert für Rauschen
                                      ;aus und Ton ein
3DAF 3E07        LD      A,07H        ;A = Registernummer
3DB1 CD323E      CALL    3E32H        ;PSG-Register A mit E laden
3DB4 D1          POP     DE           ;Lautstärke zurück
3DB5 F1          POP     AF           ;Kanalnummer zurück
3DB6 C608        ADD     A,08H        ;A = Kanalnummer + 8 = Register
                                      ;für die Lautstärke des
                                      ;gewünschten Kanals
3DB8 CD323E      CALL    3E32H        ;Lautstärke setzen
3DBB CF          RST     08H          ;Klammer geschlossen ?
3DBC 29          DEFB    ')'
3DBD C9          RET

; Notenwert = 0

3DBE CF          RST     08H          ;Danach muß ',' folgen
3DBF 2C          DEFB    ','
3DC0 CD1C2B      CALL    2B1CH        ;Lautstärke holen
3DC3 FE11        CP      11H          ;Lautstärke > 16 ?
3DC5 D24A1E      JP      NC,1E4AH     ;Ja: FC-Error
3DC8 F1          POP     AF           ;Nein: Oktavnummer zurück
3DC9 F1          POP     AF           ;Kanalnummer zurück
3DCA 1E00        LD      E,00H        ;und die Lautstärke des an-
3DCC C3B63D      JP      3DB6H        ;gegebenen Kanals auf 0 setzen
```

```
3D8B 21CF3D        LD      HL,3DCFH      ;HL = Startadresse der PLAY-
                                         ;Tabelle
3D8E 19            ADD     HL,DE         ;Notenwert * 2 addieren
3D8F 5E            LD      E,(HL)        ;und den 16-Bit Wert der
3D90 23            INC     HL            ;Tabelle entnehmen
3D91 56            LD      D,(HL)
3D92 E1            POP     HL            ;PTZ zurück
3D93 C1            POP     BC            ;B = Oktavnummer
3D94 05            DEC     B             ;Oktavnummer = 1 ?
3D95 2806          JR      Z,3D9DH       ;Ja: Werte lassen
3D97 CB3A          SRL     D             ;Nein: DE rechts schieben
3D99 CB1B          RR      E
3D9B 10FA          DJNZ    3D97H         ;DE = DE / (2 * Oktavwert)
3D9D F1            POP     AF            ;Kanalnummer -1 nach A
3D9E F5            PUSH    AF            ;und wieder retten
3D9F CB27          SLA     A             ;A = Kanalnummer * 2
3DA1 3C            INC     A             ; + 1
3DA2 CD2A3E        CALL    3E2AH         ;PSG-Register A-1 auf E, PSG-
                                         ;Register A auf D setzen
                                         ;(A zeigt auf die Register, die
                                         ;die Schwingungsdauer bestimmen)
3DA5 CD1C2B        CALL    2B1CH         ;Lautstärkenwert holen
3DA8 FE11          CP      11H           ;Lautstärke > 16 ?
3DAA 30BC          JR      NC,3D68H      ;Ja: FC-Error
3DAC D5            PUSH    DE            ;Nein: Lautstärke retten
3DAD 1E38          LD      E,38H         ;E = Programmierwert für Rauschen
                                         ;aus und Ton ein
3DAF 3E07          LD      A,07H         ;A = Registernummer
3DB1 CD323E        CALL    3E32H         ;PSG-Register A mit E laden
3DB4 D1            POP     DE            ;Lautstärke zurück
3DB5 F1            POP     AF            ;Kanalnummer zurück
3DB6 C608          ADD     A,08H         ;A = Kanalnummer + 8 = Register
                                         ;für die Lautstärke des
                                         ;gewünschten Kanals
3DB8 CD323E        CALL    3E32H         ;Lautstärke setzen
3DBB CF            RST     08H           ;Klammer geschlossen ?
3DBC 29            DEFB    ')'
3DBD C9            RET

; Notenwert = 0

3DBE CF            RST     08H           ;Danach muß ',' folgen
3DBF 2C            DEFB    ','
3DC0 CD1C2B        CALL    2B1CH         ;Lautstärke holen
3DC3 FE11          CP      11H           ;Lautstärke > 16 ?
3DC5 D24A1E        JP      NC,1E4AH      ;Ja: FC-Error
3DC8 F1            POP     AF            ;Nein: Oktavnummer zurück
3DC9 F1            POP     AF            ;Kanalnummer zurück
3DCA 1E00          LD      E,00H         ;und die Lautstärke des an-
3DCC C3B63D        JP      3DB6H         ;gegebenen Kanals auf 0 setzen
```

```
; PLAY-Tabelle
; Enthält für die 29 wählbaren Noten den entsprechenden 16-Bit Wert
; für die Schwingungsdauer-Register

  3DCF 00 00
  3DD1 5D 0D
  3DD3 E7 0B
  3DD5 9B 0A
  3DD7 02 0A
  3DD9 EB 08
  3DDB F2 07
  3DDD 14 07
  3DDF 9C 0C
  3DE1 3C 0B
  3DE3 73 09
  3DE5 6B 08
  3DE7 80 07
  3DE9 5D 0D
  3DEB 5D 0D
  3DED 5D 0D
  3DEF 4A 09
  3DF1 90 10
  3DF3 C0 0E
  3DF5 24 0D
  3DF7 68 0C
  3DF9 0C 0B
  3DFB D8 09
  3DFD C8 08
  3DFF A0 0F
  3E01 EB 0D
  3E03 B4 0B
  3E05 70 0A
  3E07 4A 09
  3E09 48 08

  3E0B F6B7              OR      0B7H         ;--
  3E0D C9                RET

; UPRO für ? (unbenutzt)
; DE nach X übergeben. Im INT-Format wenn DE > 0 sonst im SNG-Format

  3E0E CB7A              BIT     07H,D        ;D.7 = 0 ?
  3E10 2813              JR      Z,3E25H      ;Ja: DE als INT nach X übergeben
  3E12 CDEF0A            CALL    0AEFH        ;VT auf SNG setzen
  3E15 7A                LD      A,D          ; A <- D <- E <- 00H
  3E16 53                LD      D,E          ;(DE 8 Bit nach links schieben
  3E17 1E00              LD      E,00H        ;mit Überlauf in A)
  3E19 B7                OR      A            ;CY = 0
  3E1A 1F                RRA                  ;ADE wieder 1 Bit nach rechts
  3E1B CB1A              RR      D            ;schieben
  3E1D CB1B              RR      E
  3E1F 0631              LD      B,31H        ;B = Exponent für 2 hoch 17
  3E21 CD6909            CALL    0969H        ;ADE in Fließkommaformat
                                              ;umwandeln und in X ablegen
  3E24 C9                RET
```

```
; DE als INT nach X übergeben (unbenutzt)

  3E25 EB              EX      DE,HL        ;HL = DE
  3E26 CD9A0A          CALL    0A9AH        ;HL als INT nach X übergeben
  3E29 C9              RET

; UPRO zur PSG-Programmierung
; I: A  = Registernummer
;    DE = 16-Bit Wert der in die Register A und A-1 übergeben wird
; (Entspricht den Befehlen SOUND A,D und SOUND A-1,E

  3E2A 47              LD      B,A          ;B = Registernummer
  3E2B D3F8            OUT     (0F8H),A     ;Register A anwählen
  3E2D 7A              LD      A,D          ;und D
  3E2E D3F9            OUT     (0F9H),A     ;übergeben
  3E30 05              DEC     B  -         ;B = Registernummer -1
  3E31 78              LD      A,B          ;A = B
  3E32 D3F8            OUT     (0F8H),A     ;Register A(-1) anwählen
  3E34 7B              LD      A,E          ;und E
  3E35 D3F9            OUT     (0F9H),A     ;übergeben
  3E37 C9              RET

; PAINT

  3E38 AF              XOR     A            ;A = 0
  3E39 2B              DEC     HL           ;PTZ -1
  3E3A 3C              INC     A            ;A +1 (Zähler für Anzahl der
                                            ;angegebenen Werte
                                            ;(zwischen 3 und 5)
  3E3B 08              EX      AF,AF'       ;A' = Zähler
  3E3C D7              RST     10H          ;PTZ erhöhen
  3E3D CD1C2B          CALL    2B1CH        ;Wert holen
  3E40 F5              PUSH    AF           ;Wert retten
  3E41 2B              DEC     HL           ;PTZ -1
  3E42 D7              RST     10H          ;PTZ erhöhen
                                            ;Befehlsende erreicht ?
  3E43 280C            JR      Z,3E51H      ;Ja: weiter bei 3E51H
  3E45 FE2C            CP      2CH          ;Nein: Komma als Trennzeichen ?
  3E47 2005            JR      NZ,3E4EH     ;Nein: SN-Error
  3E49 08              EX      AF,AF'       ;A = Zähler
  3E4A FE05            CP      05H          ;Mehr als 5 Werte ?
  3E4C 38EC            JR      C,3E3AH      ;Nein: nächsten Wert holen
  3E4E C39719          JP      1997H        ;Ja: SN-Error

  3E51 08              EX      AF,AF'       ;A = Zähler
  3E52 FE03            CP      03H          ;weniger als 3 Werte ?
  3E54 38F8            JR      C,3E4EH      ;Ja: SN-Error
  3E56 3D              DEC     A            ;A = Zähler -2
  3E57 3D              DEC     A
  3E58 321D43          LD      (431DH),A    ;Zähler -2 (entspricht der Anzahl
                                            ;der angegebenen Begrenzungs-
                                            ;farben) abspeichern
```

```
3E5B 111843        LD     DE,4318H      :DE = Anfangsadresse der
                                        :Begrenzungsfarbentabelle -1
3E5E 47            LD     B,A           :B = Anzahl der B.-Farben
3E5F 83            ADD    A,E           :+ LSB von DE
3E60 5F            LD     E,A           :DE = Ende der Tabelle
3E61 F1            POP    AF            :Wert aus Stack holen
3E62 3D            DEC    A             :-1
3E63 FE04          CP     04H           :Angegebener Farbwert > 4 ?
3E65 D24A1E        JP     NC,1E4AH      :Ja: FC-Error
3E68 12            LD     (DE),A        :Nein: Wert abspeichern
3E69 1B            DEC    DE            :Zeiger -1
3E6A 10F5          DJNZ   3E61H         :Nächsten Wert
3E6C F1            POP    AF            :Y-Koordinate nach A
3E6D FE66          CP     66H           :Y > 101 ?
3E6F 30F4          JR     NC,3E65H      :Ja: FC-Error
3E71 57            LD     D,A           :Nein: D = Y-Koordinate
3E72 F1            POP    AF            :X-Koordinate nach A
3E73 FEA0          CP     0A0H          :X > 159 ?
3E75 30EE          JR     NC,3E65H      :Ja: FC-Error
3E77 5F            LD     E,A           :Nein: E = X-Koordinate
3E78 E5            PUSH   HL            :PTZ retten
3E79 21FFFF        LD     HL,0FFFFH     :HL = -1 (Flag für Routinenende)
3E7C E5            PUSH   HL            :Flag retten
3E7D EB            EX     DE,HL         :HL = Koordinaten

: Nächsten Punkt setzen

3E7E 221E43        LD     (431EH),HL    :Koordinaten abspeichern
3E81 CD043F        CALL   3F04H         :Farbe setzen, wenn Begrenzung
                                        :noch nicht erreicht
3E84 2806          JR     Z,3E8CH       :weiter bei 3E8CH wenn Begrenzung
                                        :schon erreicht
3E86 2D            DEC    L             :Sonst X -1
3E87 3EFF          LD     A,0FFH        :A = -1
3E89 BD            CP     L             :X = -1 ?
3E8A 20F5          JR     NZ,3E81H      :Nein: Nächsten Punkt
3E8C 2C            INC    L             :Ja: X +1
3E8D 7D            LD     A,L           :A = X-Koordinate
3E8E 322043        LD     (4320H),A     :X abspeichern (linker Rand)
3E91 2A1E43        LD     HL,(431EH)    :Letzte Koordinaten nach HL
3E94 2C            INC    L             :X +1
3E95 3EA0          LD     A,0A0H        :A = 160
3E97 BD            CP     L             :X = 160 ?
3E98 2805          JR     Z,3E9FH       :Ja: weiter bei 3E9FH
3E9A CD043F        CALL   3F04H         :Nein: Farbe setzen wenn
                                        :Begrenzung noch nicht erreicht
3E9D 20F5          JR     NZ,3E94H      :Nächsten Punkt wenn Begrenzung
                                        :noch nicht erreicht
3E9F 2D            DEC    L             :X -1
3EA0 7D            LD     A,L           :A = X-Koordinate
3EA1 322143        LD     (4321H),A     :X abspeichern (rechter Rand)
```

```
3EA4 3A1F43            LD      A,(431FH)    ;A = Y-Koordinate
3EA7 A7                AND     A            ;Y = 0 ?
3EA8 280C              JR      Z,3EB6H      ;Ja: weiter bei 3EB6H
3EAA 3D                DEC     A            ;Y -1
3EAB 67                LD      H,A          ;H = Y-Koordinate
3EAC CDC33E            CALL    3EC3H        ;Y-Koordinate überprüfen
3EAF 3A1F43            LD      A,(431FH)    ;A = Y-Koordinate
3EB2 FE65              CP      65H          ;Y = 102 ?
3EB4 2805              JR      Z,3EBBH      ;Ja: weiter bei 3EBBH
3EB6 3C                INC     A            ;Nein: Y +1
3EB7 67                LD      H,A          ;H = Y-Koordinate
3EB8 CDC33E            CALL    3EC3H        ;Y-Koordinate überprüfen
3EBB 3EFF              LD      A,0FFH       ;A = -1
3EBD E1                POP     HL           ;Y-Koordinate zurück
3EBE BC                CP      H            ;Flag erreicht ?
3EBF 20BD              JR      NZ,3E7EH     ;Nein: Nächsten Punkt
3EC1 E1                POP     HL           ;Ja: PTZ zurück
3EC2 C9                RET

; Y-Koordinate überprüfen
; I: H = Y-Koordinate

3EC3 0EFF              LD      C,0FFH       ;C = 255 (Flag)
3EC5 3A2043            LD      A,(4320H)    ;A = X-Koordinate des linken
                                            ;Randes
3EC8 6F                LD      L,A          ;L = Linker Rand
3EC9 3A2143            LD      A,(4321H)    ;A = X-Koordinate des rechten
                                            ;Randes
3ECC 95                SUB     L            ;A = Abstand zwischen beiden
                                            ;Rändern
3ECD 47                LD      B,A          ;B = Abstand
3ECE 04                INC     B            ; +1
3ECF C8                RET     Z            ;Fertig wenn beide Ränder
                                            ;identisch sind

; Gesamte Zeile der neuen Y-Koordinate überprüfen

3ED0 C5                PUSH    BC           ;Register retten
3ED1 E5                PUSH    HL
3ED2 CDF13E            CALL    3EF1H        ;Begrenzungsfarbe bei L,H
                                            ;erreicht ?
3ED5 E1                POP     HL           ;Register zurück
3ED6 C1                POP     BC
3ED7 2006              JR      NZ,3EDFH     ;Nein: weiter bei 3EDFH
3ED9 0EFF              LD      C,0FFH       ;Ja: Flag = 255
3EDB 2C                INC     L            ;X +1
3EDC 10F2              DJNZ    3ED0H        ;nächste X-Koordinate kontrol-
                                            ;lieren
3EDE C9                RET
```

```
; Begrenzungsfarbe an der neuen Y-Koordinate noch nicht erreicht

  3EDF AF              XOR    A           ;A = 0
  3EE0 B9              CP     C           ;Flag = 0 ?
                                          ;(Wurde dieser Bereich schon
                                          ;erkannt ?)
  3EE1 28F8            JR     Z,3EDBH     ;Ja: weiter bei 3EDBH
  3EE3 0E02            LD     C,02H       ;Nein: C = 2 (für 1963H)
  3EE5 C5              PUSH   BC          ;BC retten
  3EE6 CD6319          CALL   1963H       ;Noch Platz im Stack ?
  3EE9 C1              POP    BC          ;BC zurück
  3EEA D1              POP    DE          ;RET-Adr nach DE
  3EEB E5              PUSH   HL          ;X,Y-Koordinaten retten
  3EEC D5              PUSH   DE          ;RET-Adr wieder ins Stack
  3EED 0E00            LD     C,00H       ;Flag = 0
  3EEF 18EA            JR     3EDBH       ;weiter bei 3EDBH

; Begrenzungsfarbe am Punkt L,H erreicht ?
; Ja: Z, Nein: NZ

  3EF1 CD3A3F          CALL   3F3AH       ;A = CPOINT( L,H )
  3EF4 E5              PUSH   HL          ;Koordinaten retten
  3EF5 211D43          LD     HL,431DH    ;HL : Anzahl der Begrenzungs-
                                          ;farben
  3EF8 46              LD     B,(HL)      ;B = Anzahl der Beg.-Farben
  3EF9 211943          LD     HL,4319H    ;HL = Zeiger auf Begrenzungs-
                                          ;farbentabelle
  3EFC BE              CP     (HL)        ;Begrenzungsfarbe erreicht ?
  3EFD 2803            JR     Z,3F02H     ;Ja: Fertig
  3EFF 23              INC    HL          ;Zeiger +1
  3F00 10FA            DJNZ   3EFCH       ;Nächste Farbe prüfen
  3F02 E1              POP    HL          ;Koordinaten zurück
  3F03 C9              RET

; Punkt L,H in neue Farbe umwandeln, falls Begrenzungsfarbe noch nicht erreicht

  3F04 E5              PUSH   HL          ;Koordinaten retten
  3F05 CDF13E          CALL   3EF1H       ;Begrenzungsfarbe erreicht ?
  3F08 F5              PUSH   AF          ;Flags retten
  3F09 41              LD     B,C         ;B = X MOD 4 (von CPOINT)
  3F0A 3A1943          LD     A,(4319H)   ;A = neue Farbe
  3F0D 4F              LD     C,A         ;C = A
  3F0E C4B03B          CALL   NZ,3BB0H    ;Neue Farbe setzen, falls
                                          ;Begrenzungsfarbe noch nicht
                                          ;erreicht
  3F11 F1              POP    AF          ;Flags zurück
  3F12 E1              POP    HL          ;Koordinaten zurück
  3F13 C9              RET

; Alte KEYPAD1-Routine (jetzt unbenutzt)

  3F14 16FE            LD     D,0FEH
  3F16 1802            JR     3F1AH
```

```
: Alte KEYPAD2-Routine (jetzt unbenutzt)

  3F18 16F7              LD      D.0F7H
  3F1A CD873A            CALL    3A87H
  3F1D 6F                LD      L.A
  3F1E 2600              LD      H.00H
  3F20 C9                RET

: UPRO für die Ausdrucksbearbeitung (ab 2337H)
: Hier Fortsetzung von 3F7AH
: Abfangen von COLOUR-Basic Funktionen

  3F21 FE91              CP      91H         :'SOUND'-Token ?
  3F23 CAF436            JP      Z.36F4H     :Ja: weiter bei 36F4H
  3F26 FE8A              CP      8AH         :'SCALE'-Token ?
  3F28 CA0A37            JP      Z.370AH     :Ja: weiter bei 370AH
  3F2B FE80              CP      80H         :'COLOUR'-Token ?
  3F2D CA6136            JP      Z.3661H     :Ja: weiter bei 3661H
  3F30 C30525            JP      2505H       :Nein: zurück zur Ausdrucks-
                                             :bearbeitung

: VERIFY

  3F33 23                INC     HL          :PTZ +1
  3F34 C3292C            JP      2C29H       :weiter bei 2C29H

  3F37 6E                LD      L.(HL)      :--
  3F38 63                LD      H.E
  3F39 77                LD      (HL).A

: UPRO für CPOINT und PAINT
: A = CPOINT ( L , H ) (AF.BC.DE.HL)
: I: L = X-Koordinate des zu testenden Punktes
:    H = Y-Koordinate des zu testenden Punktes
: O: A = Farbwert des Punktes

  3F3A 7D                LD      A.L         :A = X-Koordinate
  3F3B 6C                LD      L.H         :HL = Y-Koordinate
  3F3C 2600              LD      H.00H       :MSB von HL auf 0
  3F3E 54                LD      D.H         :DE = HL
  3F3F 5D                LD      E.L
  3F40 29                ADD     HL.HL       :HL = Y *  2
  3F41 29                ADD     HL.HL       :HL = Y *  4
  3F42 19                ADD     HL.DE       :HL = Y *  5
  3F43 29                ADD     HL.HL       :HL = Y * 10
  3F44 29                ADD     HL.HL       :HL = Y * 20
  3F45 29                ADD     HL.HL       :HL = Y * 40
  3F46 5F                LD      E.A         :E = X-Koordinate
  3F47 CB3B              SRL     E           :E = X * 2
  3F49 CB3B              SRL     E           :E = X * 4
```

```
3F4B 1648           LD      D,48H       ;DE = Startadresse des Grafik-
                                        ;speichers + 4 * X (entspricht
                                        ;der 'Spalte' des zu setzenden
                                        ;Punktes)
3F4D 19             ADD     HL,DE       ;Y * 40 ('Zeilenposition') hinzu-
                                        ;addieren = Adresse
3F4E E603           AND     03H         ;A = X MOD 3
3F50 3C             INC     A           ;A +1
3F51 4F             LD      C,A         ;C = A (für PAINT)
3F52 47             LD      B,A         ;B = A als Zähler
3F53 7E             LD      A,(HL)      ;Alten Wert holen
3F54 07             RLCA                ;Gewünschten Punkt in Bit-
3F55 07             RLCA                ;positionen 0 und 1
3F56 10FC           DJNZ    3F54H       ;schieben
3F58 E603           AND     03H         ;Bits 2 bis 7 löschen = Farb-
3F5A C9             RET      -          ;wert des gewünschten Punktes

3F5B 00             NOP                 ;--

: CPOINT

3F5C CF             RST     08H         ;Klammer auf ?
3F5D 28
3F5E CD1C2B         CALL    2B1CH       ;X-Koordinate holen
3F61 F5             PUSH    AF          ;und retten
3F62 CF             RST     08H         ;Komma als Trennung ?
3F63 2C             DEFB    ','
3F64 CD1C2B         CALL    2B1CH       ;Y-Koordinate holen
3F67 F5             PUSH    AF          ;und retten
3F68 CF             RST     08H         ;Klammer zu ?
3F69 29             DEFB    ')'
3F6A D1             POP     DE          ;Y-Koordinate nach D
3F6B F1             POP     AF          ;X-Koordinate nach A
3F6C EB             EX      DE,HL       ;DE = PTZ, H = Y
3F6D D5             PUSH    DE          ;PTZ retten
3F6E 6F             LD      L,A         ;L = X
3F6F CD3A3F         CALL    3F3AH       ;A = CPOINT ( L , H )
3F72 6F             LD      L,A         ;HL = Farbwert
3F73 2600           LD      H,00H       ;MSB auf 0 setzen
3F75 CD9A0A         CALL    0A9AH       ;HL als INT nach X übergeben
3F78 E1             POP     HL          ;PTZ zurück
3F79 C9             RET

: UPRO für die Ausdrucksbearbeitung (siehe 2501H)
: Abfangen von COLOUR-Basic Funktionen

3F7A CAFE27         JP      Z,27FEH     ;Sprung nach 27FEH wenn
                                        ;'USR'-Token gefunden
3F7D FEFF           CP      0FFH        ;COLOUR-Basic Token ?
3F7F C20425         JP      NZ,2504H    ;Nein: weiter bei 2504H
3F82 D7             RST     10H         ;Ja: PTZ erhöhen, nächstes Token
                                        ;nach A
```

```
 3F83 23              INC    HL          ;PTZ +1
 3F84 FE82            CP     82H         ;'KEYPAD'-Token ?
 3F86 CA0F3A          JP     Z,3A0FH     ;Ja: weiter bei 3A0FH
 3F89 FE8F            CP     8FH         ;'CPOINT'-Token ?
 3F8B 28CF            JR     Z,3F5CH     ;Ja: weiter bei 3F5CH
 3F8D FE83            CP     83H         ;'JOY'-Token ?
 3F8F CA1F3A          JP     Z,3A1FH     ;Ja: weiter bei 3A1FH
 3F92 C3213F          JP     3F21H       ;Nein: weiter bei 3F21H

; SOUND

 3F95 CD1C2B          CALL   2B1CH       ;Registernummer holen
 3F98 FE10            CP     10H         ;Registernummer > 15 ?
 3F9A D24A1E          JP     NC,1E4AH    ;Ja: FC-Error
 3F9D F5              PUSH   AF          ;Nein: Registernummer retten
 3F9E CF              RST    08H         ;Komma als Trennung ?
 3F9F 2C              DEFB   ','
 3FA0 CD1C2B          CALL   2B1CH       ;Wert holen
 3FA3 5F              LD     E,A         ;E = Wert
 3FA4 F1              POP    AF          ;Registernummer zurück
 3FA5 C3323E          JP     3E32H       ;und E ins Register A übertragen

; CHAR

 3FA8 CDC23F          CALL   3FC2H       ;Wert -1 nach A holen
 3FAB FE04            CP     04H         ;Wert > 4 ?
 3FAD D24A1E          JP     NC,1E4AH    ;Ja: FC-Error
 3FB0 E603            AND    03H         ;Nein: Auf 0 bis 3 begrenzen
 3FB2 07              RLCA               ;und in Bits 3 und 4 schieben
 3FB3 07              RLCA               ;(siehe Belegung Port 255 im
 3FB4 07              RLCA               ;Handbuch S. 122)
 3FB5 47              LD     B,A         ;B = Wert
 3FB6 3A1C43          LD     A,(431CH)   ;A = Letzter Wert von Port 255
 3FB9 E6E7            AND    0E7H        ;Bits 3 und 4 auf 0 setzen
 3FBB B0              OR     B           ;Neue Bits einblenden
 3FBC D3FF            OUT    (0FFH),A    ;nach Port 255 schreiben
 3FBE 321C43          LD     (431CH),A   ;und neuen Wert abspeichern
 3FC1 C9              RET

; Ausdruck ab (HL) bearbeiten und Wert -1 in A zurückgeben
; I: HL = PTZ auf BASIC-Ausdruck
; O: A  = Wert des Ausdrucks -1

 3FC2 2B              DEC    HL          ;PTZ -1
 3FC3 D7              RST    10H         ;PTZ erhöhen
 3FC4 CD1C2B          CALL   2B1CH       ;Wert holen
 3FC7 3D              DEC    A           ;Wert -1
 3FC8 C9              RET
```

```
; Zeichen- und Cursorfarbe setzen (unbenutzt)
; Jetzt ab 35EEH ff

  3FC9 CD7A30          CALL    307AH
  3FCC 23              INC     HL
  3FCD CD7A30          CALL    307AH
  3FD0 2B              DEC     HL
  3FD1 C9              RET

; UPRO für LIST (siehe 2B91H)
; Kein Token gefunden. Textkonstante ausgeben ?

  3FD2 FE22            CP      22H          ;Textkonstante ?
  3FD4 C2892B          JP      NZ,2B89H     ;Nein: zurück nach LIST
  3FD7 03              INC     BC           ;Ja: Bufferzeiger +1
  3FD8 15              DEC     D            ;Zähler -1
  3FD9 C8              RET     Z            ;Fertig wenn Zeilenbuffer voll
  3FDA 7E              LD      A,(HL)       ;Zeichen holen
  3FDB FE22            CP      22H          ;Ende des Textes erreicht ?
  3FDD 23              INC     HL           ;Textzeiger +1
  3FDE 02              LD      (BC),A       ;Zeichen im Buffer ablegen
  3FDF CA892B          JP      Z,2B89H      ;Ja: zurück nach LIST
  3FE2 18F3            JR      3FD7H        ;Nein: nächstes Zeichen

; BGRD

  3FE4 2B              DEC     HL           ;PTZ -1
  3FE5 D7              RST     10H          ;PTZ erhöhen
  3FE6 0604            LD      B,04H        ;B,3 = 1 (für Port 255)
  3FE8 CABD38          JP      Z,38BDH      ;weiter bei 38BDH wenn kein
                                            ;Argument angegeben
  3FEB CDC23F          CALL    3FC2H        ;Argument holen
  3FEE FE04            CP      04H          ;Argument > 4 ?
  3FF0 D24A1E          JP      NC,1E4AH     ;Ja: FC-Error
  3FF3 E603            AND     03H          ;Bits 2 bis 7 löschen
  3FF5 0F              RRCA                 ;Restliche Bits nach
  3FF6 0F              RRCA                 ;Bit 6 und 7 schieben
  3FF7 47              LD      B,A          ;B = Wert
  3FF8 3A1C43          LD      A,(431CH)    ;A = letzter Portwert
  3FFB E63F            AND     3FH          ;Bits 6 und 7 löschen
  3FFD C3C238          JP      38C2H        ;B einblenden und nach Port 255
                                            ;schreiben
```

```
; RAM-Listing für das COLOUR-GENIE

 16384  4000H C3961C JP 1C96H ;RST 08H Vektor
 16387  4003H C3781D JP 1D78H ;RST 10H Vektor
 16390  4006H C3901C JP 1C90H ;RST 18H Vektor
 16393  4009H C3D925 JP 25D9H ;RST 20H Vektor
 16396  400CH C9     RET      ;RST 28H Vektor (Break-Vektor)
 16397  400DH 00     NOP
 16398  400EH 00     NOP
 16399  400FH C9     RET      ;RST 30H Vektor
 16400  4010H 00     NOP
 16401  4011H 00     NOP
 16402  4012H FB     EI       ;RST 38H Vektor (NMI-Vektor)
 16403  4013H C9     RET
 16404  4014H 00     NOP

; Tastatur DCB

 16405  4015H 01     ;DCB-Typ
 16406  4016H E303   ;Adresse der DCB-Routine: 03E3H
 16408  4018H 00     ;Flag für CTRL und MOD SEL
                     ;Bit7 = 1: CTRL gedrückt
                     ;Bit6    : MOD SEL-Flag
                               ( 1 = Grafik, 0 = ASCII-Zeichen)
 16409  4019H 07     ;2 Bytes zur Cursorprogrammierung (siehe Handbuch S. 114)
 16410  401AH 40     ;Diese Bytes werden in die Register 10 und 11 des CRTCs
                     ;kopiert
 16411  401BH 20     ;--
 16412  401CH 49     ;--

; Bildschirm DCB

 16413  401DH 07     ;DCB-Typ
 16414  401EH E430   ;Adresse der DCB-Routine: 30E4H
 16416  4020H 0044   ;Cursor-Adresse (4400H = Start des Bildschirmspeichers)
 16418  4022H 01     ;ASCII-Wert des Zeichens an der Cursorposition
                     ;(= 00H wenn Cursor ausgeschaltet)
 16419  4023H 01     ;Aktueller COLOUR-Wert
 16420  4024H 03     ;ASCII-Wert der zuletzt gedrückten Taste (für REPEAT)

; Drucker DCB

 16421  4025H 06     ;DCB-Typ
 16422  4026H E704   ;Adresse der DCB-Routine: 04E7H
 16424  4028H 43     ;Maximale Anzahl der Zeilen pro Seite + 1
 16425  4029H 00     ;Anzahl der bereits gedruckten Zeilen auf der jetzigen
                     ;Seite
 16426  402AH 00     ;--
 16427  402BH 50     ;--
 16428  402CH 42     ;--
```

```
: DOS

 16429   402DH C30050 JP 5000H
 16432   4030H C7     RST 00H
 16433   4031H 00     NOP
 16434   4032H 00     NOP

: Ansprung bei falschem DCB-Typ

 16435   4033H 3E00   LD A,00H
 16437   4035H C9     RET

: Zwischenspeicher für Tastaturroutine
: Hier werden bei jeder Tastaturabfrage die aktuellen Werte der
: acht Tastaturadressen abgelegt (siehe Handbuch S. 125)

 16438   4036H 00     : F801H
 16439   4037H 00     : F802H
 16440   4038H 00     : F804H
 16441   4039H 00     : F808H
 16442   403AH 00     : F810H
 16443   403BH 00     : F820H
 16444   403CH 00     : F840H
 16445   403DH 00     : F880H

 16446   403EH 00     :--
 16447   403FH 00
 16448   4040H 00
 16449   4041H 00
 16450   4042H 00
 16451   4043H 00
 16452   4044H 00
 16453   4045H 00
 16454   4046H 00
 16455   4047H 00
 16456   4048H 00
 16457   4049H 00
 16458   404AH 00
 16459   404BH 00
 16460   404CH 00
 16461   404DH 00
 16462   404EH 00
 16463   404FH 00
 16464   4050H 00
 16465   4051H 00
 16466   4052H 00
 16467   4053H 00
 16468   4054H 00
 16469   4055H 00
 16470   4056H 00
 16471   4057H 00
 16472   4058H 00
```

```
16473   4059H 00
16474   405AH 00
16475   405BH 00
16476   405CH 00
16477   405DH 00
16478   405EH 00
16479   405FH 00
16480   4060H 00
16481   4061H 00
16482   4062H 00
16483   4063H 00
16484   4064H 00
16485   4065H 00
16486   4066H 00
16487   4067H 00
16488   4068H 00
16489   4069H 00
16490   406AH 00
16491   406BH 00
16492   406CH 00
16493   406DH 00
16494   406EH 00
16495   406FH 00
16496   4070H 00
16497   4071H 00
16498   4072H 00
16499   4073H 00
16500   4074H 00
16501   4075H 00
16502   4076H 00
16503   4077H 00
16504   4078H 00
16505   4079H 00
16506   407AH 00
16507   407BH 00
16508   407CH 00
16509   407DH 00
16510   407EH 00
16511   407FH 00
```

; UPRO für SDIV (siehe 08CAH)

```
16512   4080H D600 SUB 00H
16514   4082H 6F   LD  L,A
16515   4083H 7C   LD  A,H
16516   4084H DE00 SBC 00H
16518   4086H 67   LD  H,A
16519   4087H 78   LD  A,B
16520   4088H DE00 SBC 00H
16522   408AH 47   LD  B,A
16523   408BH 3E00 LD  A,00H
16525   408DH C9   RET
```

```
16526  408EH 4A1E   ;Adresse für den USR-Aufruf (1E4AH = FC-Error)

16528  4090H 40     ;Multiplikator für die RND-Funktion (siehe 14F0H ff)
16529  4091H E6     ;(Mantisse des Wertes 0.253514 aber mit MSB zuerst !)
16530  4092H 4D

; UPRO für INP (siehe 2AF5H)

16531  4093H DB00 IN A,(00H)
16533  4095H C9   RET

; UPRO für OUT (siehe 2AFEH)

16534  4096H D300 OUT (00H),A
16536  4098H C9   RET


16537  4099H 00     ;ASCII-Wert der zuletzt gedrückten Taste (für INKEY$)

16538  409AH 00     ;Letzter Fehlercode (für ERR)

16539  409BH 00     ;Anzahl der ausgegebenen Zeichen in der aktuellen
                    ;Druckerzeile (DPOS)

16540  409CH 00     ;Ausgabeflag (siehe 032AH ff)

16541  409DH 28     ;Anzahl der Zeichen pro Zeile auf dem Bildschirm

16542  409EH 1E     ;Höchste erreichbare Tabulatorposition auf dem Bildschirm
                    ;(für Shift-Rechtspfeil und PRINT, )

16543  409FH 00     ;--

16544  40A0H 4C     ;Zeiger auf den Start des Stringspeichers
16545  40A1H 43     ;(wird bei Start 6 auf TOPMEM - 50 gesetzt)

16546  40A2H FE     ;Aktuelle Zeilennummer:
16547  40A3H FF     ;65535 im aktiven Befehlsmodus
                    ;65534 während der MEM SIZE Abfrage
                    ;0 - 65529 während des Programmablaufs

16548  40A4H 01     ;Zeiger auf den Start des BASIC-Programms
16549  40A5H 48     ;(4801H ohne Grafik, 5801H mit Grafik)

16550  40A6H 20     ;Cursorposition in der Bildschirmzeile (für POS-Funktion)

16551  40A7H        ;Zeiger auf den Zeilenbuffer (siehe 0361H ff)
16552  40A8H        ;(Im Basic = 41E8H)

16553  40A9H        ;INPUT-Flag (= 00H bei INPUT#)

16554  40AAH        ;3 Bytes Mantisse der letzten RND-Zahl
16555  40ABH
16556  40ACH

*16555 40ABH        ;= R-Register bei RANDOM-Aufruf
```

```
16557  40ADH        :--
16558  40AEH        :DIM-Flag. Dieses Flag zeigt ob die Routine 260DH ff ein
                    :Feld anlegen (für DIM), oder die Adresse eines
                    :Feldelements errechnen soll (für VARPTR)

16559  40AFH        :Typcode der Variablen in X (VT)

16560  40B0H        :DATA-Flag. Ist diese Flag bei der Zeilencodierung
                    :ungleich 0, wird die gesamte Zeile bis zum Zeilenende
                    :bzw. bis zum Befehlsende (':') nicht codiert. Dadurch
                    :wird erreicht, daß z.B. bei einer DATA-Zeile die Zeichen
                    :nicht in Token umgewandelt werden

16561  40B1H FF     :Zeiger auf den höchsten verfügbaren Speicherplatz - 256
16562  40B2H 7E     :(= Start der SHAPE-Tabelle = Ende des Stringspeichers)

16563  40B3H B5     :Zeiger auf nächsten freien Platz in der
16564  40B4H 40     :Stringtabelle (wird der Wert größer als 40D3, dann ist
                    :die Stringtabelle voll und es wird ein ST-Error erzeugt)

; Stringtabelle:
; Hier können Vektoren für maximal 11 Strings abgelegt werden.
; Diese Tabelle wird bei der Stringaddition und einigen anderen
; Stringfunktionen benutzt.

16565  40B5H        :Zeichenkettenlänge
16566  40B6H        :Adresse im
16567  40B7H        :Speicher
16568  40B8H
16569  40B9H
16570  40BAH
16571  40BBH
16572  40BCH
16573  40BDH
16574  40BEH
16575  40BFH
16576  40C0H
16577  40C1H
16578  40C2H
16579  40C3H
16580  40C4H
16581  40C5H
16582  40C6H
16583  40C7H
16584  40C8H
16585  40C9H
16586  40CAH
16587  40CBH
16588  40CCH
16589  40CDH
16590  40CEH
16591  40CFH
16592  40D0H
16593  40D1H
16594  40D2H
```

```
; Vektor-Zwischenspeicher bei der Übernahme einer Stringkonstanten
: in den Stringspeicher

 16595   40D3H        ;Stringlänge
 16596   40D4H        ;Stringadresse
 16597   40D5H

 16598   40D6H        :Zeiger auf den letzten Strings im Stringspeicher -1
 16599   40D7H

 16600   40D8H        ;Zwischenspeicher für PTZ. Zeiger auf die Anzahl der
 16601   40D9H        ;Dimensionen beim Anlegen eines Feldes. Zeiger auf die
                      ;nächste Feldvariable beim Sortieren des Stringspeichers

*16600   40D8H        :Formatiercode für die Zahlenumwandlung (siehe 0FBDH)

 16602   40DAH        ;Zeilennummer der aktuellen DATA-Zeile
 16603   40DBH

 16604   40DCH        ;Sperrflag für Feldvariablen. Ist dieses Flag <> 0. dann
                      :sind keine Feldvariablen erlaubt (siehe FOR-Routine)

 16605   40DDH        ;STOP-Flag: Ist dieses Flag bei RETURN <> 0 wird zum
                      ;aktiven Befehlsmodus gesprungen. bei RESUME NEXT oder
                      ;RESUME ZN wird STOP ausgeführt. Dieses Flag wird bei der
                      ;Übernahme einer neuen Zeile ins Programm auf 0 gesetzt
                      ;aber von keiner Routine ungleich 0 gesetzt. es ist also
                      ;wirkungslos

 16606   40DEH        ;READ/INPUT-Flag. = 00H bei INPUT. = AFH bei READ

 16607   40DFH        ;Anprungadresse nach Einladen eines SYSTEM-Programms
 16608   40E0H        ;Adresse bei LET. Programmstart -1 bei CLEAR. PTZ bei NEXT

 16609   40E1H        ;AUTO-Flag. AUTO ist aktiviert wenn dieses Flag <> 0 ist

 16610   40E2H        ;Aktuelle Zeilennummer der AUTO-Funktion
 16611   40E3H

 16612   40E4H        ;Abstand zur nächsten Zeilennummer der AUTO-Funktion
 16613   40E5H

 16614   40E6H        ;Aktueller PTZ vor der Ausführung eines BASIC-Befehls
 16615   40E7H

 16616   40E8H        ;BASIC-Stackpointer vor der Ausführung eines BASIC-Befehls
 16617   40E9H
```

16618	40EAH	:Zeilennummer der Zeile, in der zuletzt ein Fehler erkannt
16619	40EBH	:wurde. (= ERL)
16620	40ECH	:Zeilennummer der Zeile, die zuletzt eingegeben bzw.
16621	40EDH	:editiert wurde (für '.')
16622	40EEH	:PTZ während einer Fehlerbearbeitung
16623	40EFH	
16624	40F0H	:Zeiger auf die ON ERROR GOTO Zeile
16625	40F1H	
16626	40F2H	:ON ERROR ERROR Flag
		:(<> 00H, wenn ON ERROR GOTO aktiv ist)
16627	40F3H	:Zwischenspeicher für PTZ bei der Ausdrucksbearbeitung
16628	40F4H	:Bufferadresse des Dezimalpunkts bei der Zahlenumwandlung
16629	40F5H	:Zeilennummer der zuletzt ausgeführten Zeile nach
16630	40F6H	:END, BREAK, STOP oder ERROR (für CONT)
16631	40F7H	:PTZ nach END, BREAK, STOP oder ERROR (für CONT)
16632	40F8H	
16633	40F9H	:Zeiger auf den Anfang der Variablen
16634	40FAH	:(= Endadresse des Programmtexts + 2)
16635	40FBH	:Zeiger auf den Anfang der Feldvariablen
16636	40FCH	:(= Endadresse der einfachen Variablen + 1)
16637	40FDH	:Zeiger auf den Anfang des freien Speichers
16638	40FEH	:(= Endadresse der Feldvariablen + 1)
16639	40FFH	:Zeiger auf den nächsten DATA-Wert (wie PTZ)
16640	4100H	

```
; Typcodetabelle für DEF (DEFINT, DEFSTR, DEFSNG, DEFDBL)
; Für jeden Buchstaben (A - Z) steht ein Byte mit dem Variablentyp zur
; Verfügung. (Voreinstellung steht auf SNG)
; 02 = DEFINT
; 03 = DEFSTR
; 04 = DEFSNG
; 08 = DEFDBL

 16641  4101H 04      A
 16642  4102H 04      B
 16643  4103H 04      C
 16644  4104H 04      D
 16645  4105H 04      E
 16646  4106H 04      F
 16647  4107H 04      G
 16648  4108H 04      H
 16649  4109H 04      I
 16650  410AH 04      J
 16651  410BH 04      K
 16652  410CH 04      L
 16653  410DH 04      M
 16654  410EH 04      N
 16655  410FH 04      O
 16656  4110H 04      P
 16657  4111H 04      Q
 16658  4112H 04      R
 16659  4113H 04      S
 16660  4114H 04      T
 16661  4115H 04      U
 16662  4116H 04      V
 16663  4117H 04      W
 16664  4118H 04      X
 16665  4119H 04      Y
 16666  411AH 04      Z

 16667  411BH         :TRACE-Flag: 00H = TROFF, AFH = TRON

; Zwischenspeicher X

 16668  411CH         :Unterlaufbyte

 16669  411DH         :                                           LSB)
 16670  411EH         :                                           LSB)
 16671  411FH         :                                           LSB)
 16672  4120H         :                                           LSB)
 16673  4121H         : LSB) INT    LSB) STR      LSB) SNG        LSB) DBL
 16674  4122H         : MSB)        MSB)          LSB)            LSB)
 16675  4123H         :             der Adresse   MSB)            MSB)
 16676  4124H         :             des Vektors   EXP)            EXP)

 16677  4125H         :Signflag für Arithmetik
```

```
; Zwischenspeicher Y
; Aufbau siehe X

 16678  4126H          ;Unterlaufbyte

 16679  4127H
 16680  4128H
 16681  4129H
 16682  412AH
 16683  412BH
 16684  412CH
 16685  412DH
 16686  412EH

; Zwischenspeicher für die Zahlenumwandlung (siehe 0FBDH)

 16687  412FH          ;'%' bei Feldüberlauf
 16688  4130H          ;Vorzeichen ('+', '-' oder ' ')
 16689  4131H          ;1. Stelle
 16690  4132H          ;2. Stelle usw.
 16691  4133H
 16692  4134H
 16693  4135H
 16694  4136H
 16695  4137H
 16696  4138H
 16697  4139H
 16698  413AH
 16699  413BH
 16700  413CH
 16701  413DH
 16702  413EH
 16703  413FH
 16704  4140H
 16705  4141H
 16706  4142H
 16707  4143H
 16708  4144H
 16709  4145H
 16710  4146H
 16711  4147H
 16712  4148H
 16713  4149H

; Zwischenspeicher für Mantissen bei MUL und DIV

 16714  414AH
 16715  414BH
 16716  414CH
 16717  414DH
 16718  414EH
 16719  414FH
 16720  4150H
 16721  4151H
```

```
: Vektorentabelle für DOS bzw DISK BASIC
 16722  4152H C33B01 JP 013BH ;CVI
 16725  4155H C33B01 JP 013BH ;FN
 16728  4158H C33B01 JP 013BH ;CVS
 16731  415BH C33B01 JP 013BH ;DEF
 16734  415EH C33B01 JP 013BH ;CVD
 16737  4161H C33B01 JP 013BH ;EOF
 16740  4164H C33B01 JP 013BH ;LOC
 16743  4167H C33B01 JP 013BH ;LOF
 16746  416AH C33B01 JP 013BH ;MKI$
 16749  416DH C33B01 JP 013BH ;MKS$
 16752  4170H C33B01 JP 013BH ;MKD$
 16755  4173H C33B01 JP 013BH ;CMD
 16758  4176H C33B01 JP 013BH ;TIME$
 16761  4179H C33B01 JP 013BH ;OPEN
 16764  417CH C33B01 JP 013BH ;FIELD
 16767  417FH C33B01 JP 013BH ;GET
 16770  4182H C33B01 JP 013BH ;PUT
 16773  4185H C33B01 JP 013BH ;CLOSE
 16776  4188H C33B01 JP 013BH ;LOAD
 16779  418BH C33B01 JP 013BH ;MERGE
 16782  418EH C33B01 JP 013BH ;NAME
 16785  4191H C33B01 JP 013BH ;KILL
 16788  4194H C33B01 JP 013BH ;-- (früher '&')
 16791  4197H C33B01 JP 013BH ;LSET
 16794  419AH C33B01 JP 013BH ;RSET
 16797  419DH C33B01 JP 013BH ;INSTR
 16800  41A0H C33B01 JP 013BH ;SAVE
 16803  41A3H C33B01 JP 013BH ;LINE
```

```
; BASIC-Vektoren ins DOS

 16806   41A6H C9       RET      ;Fehlerroutine  HL = Zeiger auf Fehlertext
 16807   41A7H 00                ;E = Fehlercode (siehe 19ECH)
 16808   41A8H 00

 16809   41A9H C9       RET      ;USR  HL = PTZ auf Zeile nach USR-Befehl
 16810   41AAH 00                ;(siehe 27FEH)
 16811   41ABH 00

 16812   41ACH C9       RET      ;Rücksprung in den aktiven Befehlsmodus
 16813   41ADH 00                ;(siehe 1A1CH)
 16814   41AEH 00

 16815   41AFH C9       RET      ;Zeileneingabe
 16816   41B0H 00                ;(siehe 0368H)
 16817   41B1H 00

 16818   41B2H C9       RET      ;Nach der Zwischencodeerzeugung
 16819   41B3H 00                ;HL = PTZ, DE = ZN der Zeile, CY = 1 wenn ZN
 16820   41B4H 00                ;angegeben wurde (siehe 1AA1H)

 16821   41B5H C9       RET      ;Nach der Übernahme einer neuen Zeile
 16822   41B6H 00                ;HL = DE = Zeiger auf das Programmende
 16823   41B7H 00                ;(siehe 1AECH)

 16824   41B8H C9       RET      ;Nach der Übernahme einer neuen Zeile
 16825   41B9H 00                ;HL = PTZ auf ersten Befehl der Zeile
 16826   41BAH 00                ;(siehe 1AF2H)

 16827   41BBH C9       RET      ;Clear nach Löschen der Variablen
 16828   41BCH 00                ;HL = Zeiger auf das Programmende, DE = Zeiger
 16829   41BDH 00                ;auf den Programmstart -1 (siehe 1B8CH)

 16830   41BEH C9       RET      ;Nach Beenden einer Druckerausgabe
 16831   41BFH 00                ;(siehe 2174H)
 16832   41C0H 00

 16833   41C1H C9       RET      ;Zeichenausgabe
 16834   41C2H 00                ;C = ASCII-Code des auszugebenden Zeichens
 16835   41C3H 00                ;(siehe 032CH)

 16836   41C4H C9       RET      ;Tastaturabfrage während der Programmausführung
 16837   41C5H 00                ;(siehe 0358H)
 16838   41C6H 00

 16839   41C7H C9       RET      ;RUN
 16840   41C8H 00                ;HL = PTZ auf Zeichen nach 'RUN'
 16841   41C9H 00                ;(siehe 1EA6H)

 16842   41CAH C9       RET      ;PRINT
 16843   41CBH 00                ;HL = PTZ auf Zeichen nach 'PRINT'
 16844   41CCH 00                ;(siehe 206FH)
```

```
16845  41CDH C9    RET     ;PRINT (Zahlenwert)
16846  41CEH 00            ;(siehe 20C6H)
16847  41CFH 00

16848  41D0H C9    RET     ;Beginn einer neuen Zeile
16849  41D1H 00            ;(siehe 2103H)
16850  41D2H 00

16851  41D3H C9    RET     ;PRINT, oder PRINTTAB
16852  41D4H 00            ;(siehe 2108H bzw. 2141H)
16853  41D5H 00

16854  41D6H C9    RET     ;INPUT
16855  41D7H 00            ;HL = PTZ auf Zeichen nach 'INPUT'
16856  41D8H 00            ;(siehe 219EH)

16857  41D9H C9    RET     ;MID$ links vom Gleichheitszeichen
16858  41DAH 00            ;(siehe 2AECH)
16859  41DBH 00

16860  41DCH C9    RET     ;Datenauswertung bei READ bzw. INPUT
16861  41DDH 00            ;HL = Zeiger auf Daten (DATA-Zeile bei READ)
16862  41DEH 00            ;(siehe 222DH)

16863  41DFH C9    RET     ;Abschluß von INPUT
16864  41E0H 00            ;DE = PTZ, HL = Bufferzeiger
16865  41E1H 00            ;(siehe 2278H)

16866  41E2H C9    RET     ;SYSTEM
16867  41E3H 00            ;(siehe 02B2H)
16868  41E4H 00

; Zeilenbuffer

16869  41E5H 3A    ;Markierung
16870  41E6H 00    ;für den Start
16871  41E7H 2C    ;des Zeilenbuffers
16872  41E8H       ;255 Bytes für die Ein- und Ausgabe von BASIC-Zeilen
.
.
.
17127  42E7H       ;Ende des Zeilenbuffers

17128  42E8H
17129  42E9H
17130  42EAH
17131  42EBH
17132  42ECH
17133  42EDH
17134  42EEH
17135  42EFH
```

```
; Programmierungstabelle für den CRTC im LGR-Modus
; Erklärungen siehe S. 115 im Handbuch

 17136   42F0H 01
 17137   42F1H 00
 17138   42F2H 00
 17139   42F3H 04
 17140   42F4H 07
 17141   42F5H C4
 17142   42F6H 07
 17143   42F7H A0
 17144   42F8H 1F
 17145   42F9H 19
 17146   42FAH 00
 17147   42FBH 26
 17148   42FCH 96
 17149   42FDH 34
 17150   42FEH 28
 17151   42FFH 46

; Programmierungstabelle für den CRTC im FGR-Modus
; Erklärungen siehe S. 115 im Handbuch

 17152   4300H 00
 17153   4301H 00
 17154   4302H 00
 17155   4303H 08
 17156   4304H 00
 17157   4305H 20
 17158   4306H 01
 17159   4307H 20
 17160   4308H 74
 17161   4309H 66
 17162   430AH 1F
 17163   430BH 7E
 17164   430CH 96
 17165   430DH 34
 17166   430EH 28
 17167   430FH 46

; 3 Werte für die Zeitschleifen bei den Cassettenoperationen

 17168   4310H 46
 17169   4311H 4B
 17170   4312H 69
```

17171	4313H	;FCOLOUR-Wert
17172	4314H	;SCALE-Wert
17173	4315H	;letzter X-Wert von PLOT (für PLOT TO X,Y)
17174	4316H	;letzter Y-Wert von PLOT (für PLOT TO X,Y)
17175	4317H	;Maske für SHAPE, NSHAPE und XSHAPE
17176	4318H	;(siehe 3D1EH ff)
17177	4319H	;2. Begrenzungsfarbe bei PAINT
17178	431AH	;1. Begrenzungsfarbe bei PAINT
17179	431BH	;Neue Farbe bei PAINT
17180	431CH	;Letzter Wert der an den Port 255 ausgegeben wurde
17181	431DH	;Anzahl der bei PAINT angegebenen Begrenzungsfarben
17182	431EH	;X-Wert bei PAINT
17183	431FH	;Y-Wert bei PAINT
17184	4320H	;X-Wert des linken Randes bei PAINT
17185	4321H	;X-Wert des rechten Randes bei PAINT
17186	4322H	-
17187	4323H	
17188	4324H	
17189	4325H	
17190	4326H	
17191	4327H	
17192	4328H	
17193	4329H	
17194	432AH	
17195	432BH	
17196	432CH	
17197	432DH	
17198	432EH	
17199	432FH	
17200	4330H	
17201	4331H	
17202	4332H	
17203	4333H	
17204	4334H	
17205	4335H	
17206	4336H	
17207	4337H	
17208	4338H	
17209	4339H	
17210	433AH	
17211	433BH	
17212	433CH	
17213	433DH	
17214	433EH	
17215	433FH	
17216	4340H	
17217	4341H	
17218	4342H	
17219	4343H	

```
 17220   4344H
 17221   4345H
 17222   4346H
 17223   4347H
 17224   4348H
 17225   4349H
 17226   434AH
 17227   434BH
 17228   434CH
 17229   434DH
 17230   434EH
 17231   434FH

; Tabelle der FKEY-Texte
; Für jede FKEY-Taste sind 7 Bytes reserviert
; 00H wird als RETURN interpretiert  -

 17232   4350H 4C       ;F1: 'LIST    '
 17233   4351H 49
 17234   4352H 53
 17235   4353H 54
 17236   4354H 20
 17237   4355H 20
 17238   4356H 20

 17239   4357H 52       ;F2: 'RUN     '
 17240   4358H 55
 17241   4359H 4E
 17242   435AH 20
 17243   435BH 20
 17244   435CH 20
 17245   435DH 20

 17246   435EH 41       ;F3: 'AUTO    '
 17247   435FH 55
 17248   4360H 54
 17249   4361H 4F
 17250   4362H 20
 17251   4363H 20
 17252   4364H 20

 17253   4365H 45       ;F4: 'EDIT    '
 17254   4366H 44
 17255   4367H 49
 17256   4368H 54
 17257   4369H 20
 17258   436AH 20
 17259   436BH 20
```

```
17260   436CH 52        :F5: 'RENUM  '
17261   436DH 45
17262   436EH 4E
17263   436FH 55
17264   4370H 4D
17265   4371H 20
17266   4372H 20

17267   4373H 53        :F6: 'SYSTEM' <RETURN>
17268   4374H 59
17269   4375H 53
17270   4376H 54
17271   4377H 45
17272   4378H 4D
17273   4379H 00

17274   437AH 43        :F7: 'CLOAD  '
17275   437BH 4C
17276   437CH 4F
17277   437DH 41
17278   437EH 44
17279   437FH 20
17280   4380H 20

17281   4381H 43        :F8: 'CSAVE "'
17282   4382H 53
17283   4383H 41
17284   4384H 56
17285   4385H 45
17286   4386H 20
17287   4387H 22


17288   4388H
17289   4389H
17290   438AH
17291   438BH

17292   438CH           :Zeiger auf die COLOUR-Basic Keywordtabelle
17293   438DH

17294   438EH           :Zeiger auf die COLOUR-Basic Sprungadressen-
17295   438FH           :tabelle
```

```
; Tabelle zur Anpassung der Farbcodes

 17296   4390H 03
 17297   4391H 05
 17298   4392H 02
 17299   4393H 04
 17300   4394H 06
 17301   4395H 08
 17302   4396H 01
 17303   4397H 0E
 17304   4398H 09
 17305   4399H 10
 17306   439AH 07
 17307   439BH 0B
 17308   439CH 0C
 17309   439DH 0D
 17310   439EH 0A
 17311   439FH 0F


 17312   43A0H          ;Unbenutzt bis (einschl.) 43FFH
 .
 .
 .
 17407   43FFH

 17408   4400H          ;Start des Bildschirmspeichers
```

128 = 80H : END
129 = 81H : FOR
130 = 82H : RESET
131 = 83H : SET
132 = 84H : CLS
133 = 85H : CMD
134 = 86H : RANDOM
135 = 87H : NEXT
136 = 88H : DATA
137 = 89H : INPUT
138 = 8AH : DIM
139 = 8BH : READ
140 = 8CH : LET
141 = 8DH : GOTO
142 = 8EH : RUN
143 = 8FH : IF
144 = 90H : RESTORE
145 = 91H : GOSUB
146 = 92H : RETURN
147 = 93H : REM
148 = 94H : STOP
149 = 95H : ELSE
150 = 96H : TRON
151 = 97H : TROFF
152 = 98H : DEFSTR
153 = 99H : DEFINT
154 = 9AH : DEFSNG
155 = 9BH : DEFDBL
156 = 9CH : LINE
157 = 9DH : EDIT
158 = 9EH : ERROR
159 = 9FH : RESUME
160 = A0H : OUT
161 = A1H : ON
162 = A2H : OPEN
163 = A3H : FIELD
164 = A4H : GET
165 = A5H : PUT
166 = A6H : CLOSE
167 = A7H : LOAD
168 = A8H : MERGE
169 = A9H : NAME
170 = AAH : KILL
171 = ABH : LSET
172 = ACH : RSET
173 = ADH : SAVE
174 = AEH : SYSTEM
175 = AFH : LPRINT
176 = B0H : DEF
177 = B1H : POKE
178 = B2H : PRINT
179 = B3H : CONT
180 = B4H : LIST
181 = B5H : LLIST
182 = B6H : DELETE
183 = B7H : AUTO
184 = B8H : CLEAR
185 = B9H : CLOAD
186 = BAH : CSAVE
187 = BBH : NEW
188 = BCH : TAB(
189 = BDH : TO
190 = BEH : FN
191 = BFH : USING
192 = C0H : VARPTR
193 = C1H : USR
194 = C2H : ERL
195 = C3H : ERR
196 = C4H : STRING$
197 = C5H : INSTR
198 = C6H : CHECK
199 = C7H : TIME$
200 = C8H : MEM
201 = C9H : INKEY$
202 = CAH : THEN
203 = CBH : NOT
204 = CCH : STEP
205 = CDH : +
206 = CEH : -
207 = CFH : *
208 = D0H : /
209 = D1H : Ä
210 = D2H : AND
211 = D3H : OR
212 = D4H : >
213 = D5H : =
214 = D6H : <
215 = D7H : SGN
216 = D8H : INT
217 = D9H : ABS
218 = DAH : FRE
219 = DBH : INP
220 = DCH : POS
221 = DDH : SQR
222 = DEH : RND
223 = DFH : LOG
224 = E0H : EXP
225 = E1H : COS
226 = E2H : SIN
227 = E3H : TAN

```
228 = E4H : ATN          229 = E5H : PEEK
230 = E6H : CVI          231 = E7H : CVS
232 = E8H : CVD          233 = E9H : EOF
234 = EAH : LOC          235 = EBH : LOF
236 = ECH : MKI$         237 = EDH : MKS$
238 = EEH : MKD$         239 = EFH : CINT
240 = F0H : CSNG         241 = F1H : CDBL
242 = F2H : FIX          243 = F3H : LEN
244 = F4H : STR$         245 = F5H : VAL
246 = F6H : ASC          247 = F7H : CHR$
248 = F8H : LEFT$        249 = F9H : RIGHT$
250 = FAH : MID$         251 = FBH : '
```

Befehle mit Doppeltoken (vorher jeweils 255 = FFH)

```
128 = 80H : COLOUR       129 = 81H : FCOLOU (R)
130 = 82H : KEYPAD       131 = 83H : JOY
132 = 84H : PLOT         133 = 85H : FGR
134 = 86H : LGR          135 = 87H : FCLS
136 = 88H : PLAY         137 = 89H : CIRCLE
138 = 8AH : SCALE        139 = 8BH : SHAPE
140 = 8CH : NSHAPE       141 = 8DH : XSHAPE
142 = 8EH : PAINT        143 = 8FH : CPOINT
144 = 90H : NPLOT        145 = 91H : SOUND
146 = 92H : CHAR         147 = 93H : RENUM
148 = 94H : SWAP         149 = 95H : FKEY
150 = 96H : CALL         151 = 97H : VERIFY
152 = 98H : BGRD         153 = 99H : NBGRD
```

Dec	Hex	Keyword	Dec	Hex	Keyword
251	FBH	'	207	CFH	*
205	CDH	+	206	CEH	-
208	D0H	/	214	D6H	<
213	D5H	=	212	D4H	>
217	D9H	ABS	210	D2H	AND
246	F6H	ASC	228	E4H	ATN
183	B7H	AUTO	241	F1H	CDBL
247	F7H	CHR$	239	EFH	CINT
184	B8H	CLEAR	185	B9H	CLOAD
166	A6H	CLOSE	132	84H	CLS
133	85H	CMD	179	B3H	CONT
225	E1H	COS	186	BAH	CSAVE
240	F0H	CSNG	232	E8H	CVD
230	E6H	CVI	231	E7H	CVS
136	88H	DATA	176	B0H	DEF
155	9BH	DEFDBL	153	99H	DEFINT
154	9AH	DEFSNG	152	98H	DEFSTR
182	B6H	DELETE	138	8AH	DIM
157	9DH	EDIT	149	95H	ELSE
128	80H	END	233	E9H	EOF
194	C2H	ERL	195	C3H	ERR
158	9EH	ERROR	224	E0H	EXP
163	A3H	FIELD	242	F2H	FIX
190	BEH	FN	129	81H	FOR
218	DAH	FRE	164	A4H	GET
145	91H	GOSUB	141	8DH	GOTO
143	8FH	IF	201	C9H	INKEY$
219	DBH	INP	137	89H	INPUT
197	C5H	INSTR	216	D8H	INT
170	AAH	KILL	248	F8H	LEFT$
243	F3H	LEN	140	8CH	LET
156	9CH	LINE	180	B4H	LIST
181	B5H	LLIST	167	A7H	LOAD
234	EAH	LOC	235	EBH	LOF
223	DFH	LOG	175	AFH	LPRINT
171	ABH	LSET	200	C8H	MEM
168	A8H	MERGE	250	FAH	MID$
238	EEH	MKD$	236	ECH	MKI$
237	EDH	MKS$	169	A9H	NAME
187	BBH	NEW	135	87H	NEXT
203	CBH	NOT	161	A1H	ON
162	A2H	OPEN	211	D3H	OR
160	A0H	OUT	229	E5H	PEEK
198	C6H	CHECK	177	B1H	POKE
220	DCH	POS	178	B2H	PRINT
165	A5H	PUT	134	86H	RANDOM
139	8BH	READ	147	93H	REM
130	82H	RESET	144	90H	RESTORE
159	9FH	RESUME	146	92H	RETURN
249	F9H	RIGHT$	222	DEH	RND

172 = ACH	RSET		142 = 8EH	RUN
173 = ADH	SAVE		131 = 83H	SET
215 = D7H	SGN		226 = E2H	SIN
221 = DDH	SQR		204 = CCH	STEP
148 = 94H	STOP		244 = F4H	STR$
196 = C4H	STRING$		174 = AEH	SYSTEM
188 = BCH	TAB(		227 = E3H	TAN
202 = CAH	THEN		199 = C7H	TIME$
189 = BDH	TO		151 = 97H	TROFF
150 = 96H	TRON		191 = BFH	USING
193 = C1H	USR		245 = F5H	VAL
192 = C0H	VARPTR		209 = D1H	Ä

Befehle mit Doppeltoken (vorher jeweils 255 = FFH)

152 = 98H	BGRD		150 = 96H	CALL
146 = 92H	CHAR		137 = 89H	CIRCLE
128 = 80H	COLOUR		143 = 8FH	CPOINT
135 = 87H	FCLS		129 = 81H	FCOLOU (R)
133 = 85H	FGR		149 = 95H	FKEY
131 = 83H	JOY		130 = 82H	KEYPAD
143 = 86H	LGR		153 = 99H	NBGRD
144 = 90H	NPLOT		140 = 8CH	NSHAPE
142 = 8EH	PAINT		136 = 88H	PLAY
132 = 84H	PLOT		147 = 93H	RENUM
138 = 8AH	SCALE		139 = 8BH	SHAPE
145 = 91H	SOUND		148 = 94H	SWAP
151 = 97H	VERIFY		141 = 8DH	XSHAPE

<u>Dezimal-, Hexadezimal-, ASCII-Tabelle</u>

Dez		Hex	Zeichen	Dez		Hex	Zeichen	Dez		Hex	Zeichen
0	=	00H		1	=	01H		2	=	02H	
3	=	03H		4	=	04H		5	=	05H	
6	=	06H		7	=	07H		8	=	08H	
9	=	09H		10	=	0AH		11	=	0BH	
12	=	0CH		13	=	0DH		14	=	0EH	
15	=	0FH		16	=	10H		17	=	11H	
18	=	12H		19	=	13H		20	=	14H	
21	=	15H		22	=	16H		23	=	17H	
24	=	18H		25	=	19H		26	=	1AH	
27	=	1BH		28	=	1CH		29	=	1DH	
30	=	1EH		31	=	1FH	¯	32	=	20H	
33	=	21H	!	34	=	22H	"	35	=	23H	#
36	=	24H	$	37	=	25H	%	38	=	26H	&
39	=	27H	'	40	=	28H	(	41	=	29H	)
42	=	2AH	*	43	=	2BH	+	44	=	2CH	,
45	=	2DH	-	46	=	2EH	.	47	=	2FH	/
48	=	30H	0	49	=	31H	1	50	=	32H	2
51	=	33H	3	52	=	34H	4	53	=	35H	5
54	=	36H	6	55	=	37H	7	56	=	38H	8
57	=	39H	9	58	=	3AH	:	59	=	3BH	;
60	=	3CH	<	61	=	3DH	=	62	=	3EH	>
63	=	3FH	?	64	=	40H	@	65	=	41H	A
66	=	42H	B	67	=	43H	C	68	=	44H	D
69	=	45H	E	70	=	46H	F	71	=	47H	G
72	=	48H	H	73	=	49H	I	74	=	4AH	J
75	=	4BH	K	76	=	4CH	L	77	=	4DH	M
78	=	4EH	N	79	=	4FH	O	80	=	50H	P
81	=	51H	Q	82	=	52H	R	83	=	53H	S
84	=	54H	T	85	=	55H	U	86	=	56H	V
87	=	57H	W	88	=	58H	X	89	=	59H	Y
90	=	5AH	Z	91	=	5BH	[	92	=	5CH	\
93	=	5DH	]	94	=	5EH	^	95	=	5FH	_
96	=	60H	`	97	=	61H	a	98	=	62H	b
99	=	63H	c	100	=	64H	d	101	=	65H	e
102	=	66H	f	103	=	67H	g	104	=	68H	h
105	=	69H	i	106	=	6AH	j	107	=	6BH	k
108	=	6CH	l	109	=	6DH	m	110	=	6EH	n
111	=	6FH	o	112	=	70H	p	113	=	71H	q
114	=	72H	r	115	=	73H	s	116	=	74H	t
117	=	75H	u	118	=	76H	v	119	=	77H	w
120	=	78H	x	121	=	79H	y	122	=	7AH	z
123	=	7BH	{	124	=	7CH	¦	125	=	7DH	}
126	=	7EH	~	127	=	7FH	■	128	=	80H	┓
129	=	81H	┏	130	=	82H	┛	131	=	83H	┗
132	=	84H	┫	133	=	85H	┣	134	=	86H	┻
135	=	87H	┳	136	=	88H	━	137	=	89H	┃

Dec		Hex	Dec		Hex	Dec		Hex
138	=	8AH	139	=	8BH	140	=	8CH
141	=	8DH	142	=	8EH	143	=	8FH
144	=	90H	145	=	91H	146	=	92H
147	=	93H	148	=	94H	149	=	95H
150	=	96H	151	=	97H	152	=	98H
153	=	99H	154	=	9AH	155	=	9BH
156	=	9CH	157	=	9DH	158	=	9EH
159	=	9FH	160	=	A0H	161	=	A1H
162	=	A2H	163	=	A3H	164	=	A4H
165	=	A5H	166	=	A6H	167	=	A7H
168	=	A8H	169	=	A9H	170	=	AAH
171	=	ABH	172	=	ACH	173	=	ADH
174	=	AEH	175	=	AFH	176	=	B0H
177	=	B1H	178	=	B2H	179	=	B3H
180	=	B4H	181	=	B5H	182	=	B6H
183	=	B7H	184	=	B8H	185	=	B9H
186	=	BAH	187	=	BBH	188	=	BCH
189	=	BDH	190	=	BEH	191	=	BFH
192	=	C0H	193	=	C1H	194	=	C2H
195	=	C3H	196	=	C4H	197	=	C5H
198	=	C6H	199	=	C7H	200	=	C8H
201	=	C9H	202	=	CAH	203	=	CBH
204	=	CCH	205	=	CDH	206	=	CEH
207	=	CFH	208	=	D0H	209	=	D1H
210	=	D2H	211	=	D3H	212	=	D4H
213	=	D5H	214	=	D6H	215	=	D7H
216	=	D8H	217	=	D9H	218	=	DAH
219	=	DBH	220	=	DCH	221	=	DDH
222	=	DEH	223	=	DFH	224	=	E0H
225	=	E1H	226	=	E2H	227	=	E3H
228	=	E4H	229	=	E5H	230	=	E6H
231	=	E7H	232	=	E8H	233	=	E9H
234	=	EAH	235	=	EBH	236	=	ECH
237	=	EDH	238	=	EEH	239	=	EFH
240	=	F0H	241	=	F1H	242	=	F2H
243	=	F3H	244	=	F4H	245	=	F5H
246	=	F6H	247	=	F7H	248	=	F8H
249	=	F9H	250	=	FAH	251	=	FBH
252	=	FCH	253	=	FDH	254	=	FEH
255	=	FFH						

